PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06T 7 /20

A1

(11) International Publication Number: WO 00/04508

(43) International Publication Date: 27 January 2000 (27.01.00)

(21) International Application Number: PCT/US99/16395

(22) International Filing Date: 20 July 1999 (20.07.99)

(30) Priority Data:
60/093,492 20 July 1998 (20.07.98) US

(71) Applicant: GEOMETRIX, INC. [US/US]; Patent Office, 124 Race Street, San Jose, CA 95126 (US).

(72) Inventors: ZWERN, Arthur; 2226 Coastland Avenue, San Jose, CA 95125 (US). FEJES, Sandor; 4859 Clydelle Avenue, San Jose, CA 95124 (US). CHEN, Jinlong; 4664 Cheeney Street, Santa Clara, CA 95054 (US). WAUPOTITSCH, Roman; 200 Towne Terrace, 15, Los Gatos, CA 95032 (US).

(74) Agent: ZHENG, Joe; Silicon Valley Patent Agency, 18026 King Court, Saratoga, CA 95070 (US).

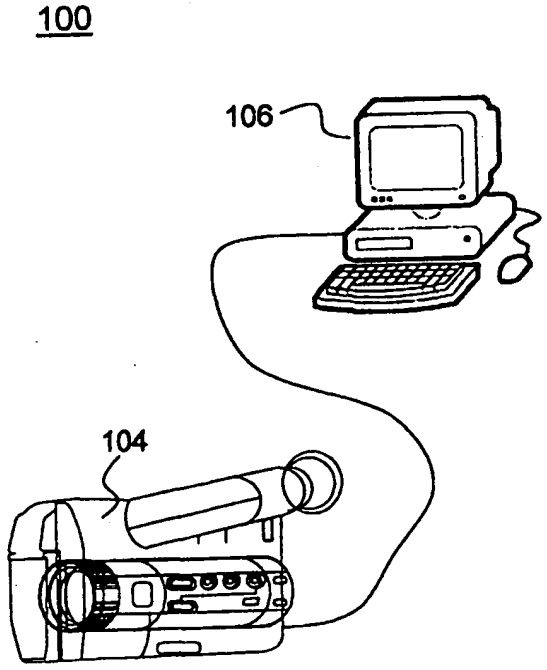
(81) Designated States: CN, JP, KR, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: AUTOMATED 3D SCENE SCANNING FROM MOTION IMAGES

(57) Abstract

A system to automatically generate a fully-textured 3D model of an object from motion images is disclosed. The system tracks salient features in the motion images based on detected salient features using a salient feature operator. A features tracking map is used to construct feature blocks comprising the tracked salient features, each of the feature blocks is then provided as input to a camera motion estimation process that is controlled to provide solutions for a perspective camera motion estimation process, dense points to be used for generating a mesh model are extracted. Finally the mesh model is textured with respect to the motion images so that a fully-textured 3D model is produced.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

Automated 3D Scene Scanning From Motion Images

CROSS-REFERENCE TO RELATED APPLICATION

This application claims the benefits of the provisional application, No. 60/093,492, filed 07/20/98, entitled " Scene Scanning Apparatus and Methods ", which is hereby incorporated by reference for all purposes.

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention generally relates to image-based motion estimation and detection systems and more particularly relates to methods and systems for modeling 3D scene from motion images produced by a video imaging system.

Description of the Related Art

The problem of determining a 3D structure of a scene from multiple images of the scene is an important branch of the photogrammetry field, and solutions have been available since the late 1800s (*see* Atkinson, K.B.; "Instrumentation For Non-Topographic

Photogrammetry"; American Society for Photogrammetry and Remote Sensing; 1989.). In simple terms, parallax-based triangulation techniques such as resection and intersection can be used to determine a selected control point's position in 3D space if that point can be located accurately and unambiguously in at least two photographs taken from two camera positions known precisely in six degrees-of-freedom (*see* Williamson, James R., and Brill, Michael H; "Dimensional Analysis Through Perspective"; Kendall/Hunt Publishing; 1990). Historically, control points are determined by a user, via selection of a salient visual feature in one image, and manual selection of the same feature in the other image(s). The process is very labor-intensive.

By using a network of multiple camera positions and multiple feature correspondences between each image, a series of simultaneous equations can be solved to determine the 3D positions of many selected control points. If the camera positions are known, the equations are linear, while if the camera positions are not known, the equations are non-linear. The latter requires extensive computation, and can be more prone to accuracy errors and stability problems. In the non-linear case, an initial approximation of at least some camera parameters is required in order to achieve convergence of a solution (*see* McGlone, Chris J.; "Analytic Data Reduction Schemes In Non-Topographic Photogrammetry"; American Society for Photogrammetry and Remote Sensing; 1989). A variety of software packages for solving various photogrammetry problems are commercially available (*see*

Ruther, H.; "An Overview of Software In Non-Topographic Photogrammetry" American Society for Photogrammetry and Remote Sensing; 1989) ranging from costly military packages to recently including consumer-priced desktop products such as "3D Builder" from 3D Construction Company; Elizabethton TN, USA and "Photomodeler" from Eos Systems Inc., Vancouver BC, Canada. All such products require the labor-intensive manual control point selection and correspondence process. Because visual features appear different from different viewing angles and computer-based matching techniques are not sufficiently sophisticated to recognize features such as a building corner in widely disparate views, there is too much ambiguity to robustly automate the feature correspondence process using a small number of camera positions.

There is therefore a great need for automated determination of 3D structure and camera motion from a series of video frames. There have been many efforts in the automated photogrammetry arena. Many use the so-called "motion factorization" method, which was first proposed in a 1990 Thesis (see Debrunner, Christian Hans; "Structure and Motion From Long Image Sequences"; Ph.D. Thesis; University of Illinois; 1990). The approach uses singular-value-decomposition to factor a large matrix of control point data into a camera motion matrix and a scene structure matrix. After a few frames of video are analyzed, a reasonable approximation of the scene's 3D structure begins to form, which is then improved within the limits of system noise by incrementally recalculating the single-value-decomposition as each

new video frame is received. However, the motion factorization approach has demonstrated several shortcomings:

1) Factorization requires that every visible feature be corresponded between every frame in the video sequence in order to completely fill the factorization matrix. This typically limits the maximum camera motion severely, and may make it difficult to use the approach for generating a single 3D model containing of all sides of an object. It also limits the density of the extracted polygonal mesh, since any visual feature which can not be corresponded between all frames of the image sequence must be ignored.

2) Factorization is highly sensitive to feature tracking errors, as even a single mis-tracked feature dramatically modifies the entire extracted 3D structure. This limits factorization to use only with the most salient features in an image sequence, resulting in sparse 3D point clouds.

3) Factorization involves significant camera model approximations and assumptions (such as orthographic, weak perspective, or paraperspective projection), which can introduce significant error outside of controlled laboratory demonstrations. Since factorization using true-perspective projection is non-linear and often fails to converge, most factorization approaches use weak-perspective. Weak perspective only yields the correct shape of an object when the object has a very small depth compared to its distance to the camera –

a situation which can only be approximated for real objects when they are ideally infinitely distant.

The factorization process is an interesting approach and provides one of the solutions in the automated photogrammetry. Nevertheless, the assumptions and conditions are too restrictive and unrealistic in view of many practical applications. Thus, it will be a desirable significant advancement if the factorization process could be used to provide practical solutions when the above limitations are overcome.

SUMMARY OF THE INVENTION

The present invention relates to techniques that provide for automatically generating fully-textured 3D models of objects from a sequence of motion images. A 3D modeling system employing the invention disclosed herein can be used to model 3D objects or targets in a wide ranges from a simple man-made part to a natural scene.

According to one aspect of the present invention, motion images are generated using a video camera or still photo camera that is moved gradually around or relatively against an object. A salient feature operator is applied to only an initial image or those images that appear to have lost some of the features being tracked. With the extracted salient features, a tracking of these salient features is carried out using multi-resolution feature structures generated for each of the salient features.

According to another aspect of the present invention, a features tracking map is used to construct feature blocks, each is then provided as input to a factorization process that is used in a feedback correction system. As the factorization process works right under orthography, results from the factorization process are used recursively to adjust image positions of the features to emulate the orthographic projections so as to derive valid camera motion segments that are then assembled to obtain the complete motion . The use of the orthographic factorization embedded in the

recursive feedback framework provides a mechanism to obtain the accurate camera motion and 3D points from a true perspective camera.

According to still another aspect of the present invention, a global optimization technique, such as a non-linear optimizing methodology, is used to refine 3D coordinates of the 3D points in accordance with the obtained camera motion so as to minimize their back-projection errors with respect to their original locations.

According to still another aspect of the present invention, a plurality of dense points are detected and then tracked using the constraints by epipolar lines in conjunction with the knowledge of the camera motion to avoid extensive detection and reduce false matches of these dense points. The 3D positions of the dense points are then estimated by triangulation. A mesh model is finally built upon the dense points by computing the 3D Delaunay triangulation.

According to yet still another aspect of the present invention, in generating texture mapping for the mesh model, a mechanism is provided to export the patches assembling the mesh model in a commonly used image file format. The patches can be subsequently modified independently with an image processing application. The texture mapping process described herein can be implemented to take advantage of the graphics accelerator architecture commonly in most computer systems. Redirecting the

graphics accelerator to draw into a buffer in memory rather than the buffer for the monitor can yield a much more efficient mapping of the textures, hence high performance of the overall system

The invention can be implemented in numerous ways, including a method, a system and a computer readable medium containing program code for automatically generating a fully-textured 3D model of an object without extensive knowledge, intensive labors and expensive equipment. The advantages of the invention are numerous. Different embodiments or implementations may yield one or more of the following unique advantages and benefits.

One of the important advantages and benefits in present invention is the use of efficient feature extraction and tracking mechanisms to track salient features in a sequence of images. The feature extraction mechanism uses a salient feature operator to accurately and unbiasedly locate salient features based on a 3D interpretation of the image intensity/color. The tracking mechanism uses multi-resolution feature structures that provide an effectively large search area yet precise location of all salient features being tracked. The tracking mechanism is capable of handling perspective distortions or other view changes of the features, reacquiring lost features when needed and fully adaptively decimating high rate video frames to reduce redundant input data while still maintaining sufficient feature correspondence.

Another one of the important advantages and benefits in present invention is the use of a factorization approach under orthography. A feedback system emulates the orthographic camera model by iteratively "correcting" the perspective camera model so that the factorization approach provides practical and accurate solutions.

Other advantages, benefits, objects and features of the present invention, together with the foregoing, are attained in the exercise of the invention in the following description and resulting in the embodiment illustrated in the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features, aspects, and advantages of the present invention will become better understood with regard to the following description, appended claims, and accompanying drawings where:

Figure 1 demonstrates a system in which the present invention may be practiced;

Figure 2 shows a block diagram of a preferred internal construction of computer system that may be used in the system of **Figure 1**;

Figure 3 illustrates a 3D drawing of an intensity image that includes a white area and a dark area;

Figure 4A shows two exemplary consecutive images and successively received from an imager;

Figure 4B shows that an exemplary multi-resolution hierarchical feature structure for extracting a feature in one of the images in **Figure 4A**;

Figure 4C shows K image structures from a single image and each of image structures is for one feature;

Figure 4D shows, as an example, what is called herein a "features tracking map", or simply features map;

Figure 4E shows a flowchart of the feature extraction process;

Figure 4F shows a series of images are receiving from the imager and images at L-th, 1L-th, 2L-th, ... frame are regularly used for feature and extraction;

Figure 4E illustrates a template update in feature tracking among a set of consecutive images;

Figure 5 shows a flowchart of a camera motion estimation process;

Figure 6A shows a features map being divided into individual feature blocks, each pair of the feature blocks overlaps;

Figure 6B shows displacement Tfq of a scene point Pfq projected onto two adjacent images;

Figure 6C shows an implementation of the camera motion estimation process using the factorization method;

Figure 6D illustrates how a cube is projected under an orthographic and respective projection, respectively;

Figures 7A-7C show, respectively, a process of combining the camera motion from a number of camera motion segments concatenated over an overlapping portion and an exemplary resultant camera motion ;

Figure 8 shows a flowchart of the depth mapping process disclosed herein;

Figure 9A illustrates a house image being detected for the line features;

Figure 9B shows a point P in the object being projected on two adjacent image planes;

Figure 9C illustrates a flowchart of generating a self-constraint and interconnected triangular mesh model based on the Delaunay triangulation;

Figure 10A shows a process flowchart of applying the texture patterns to a mesh model;

Figure 10B shows a flowchart of the textured patch generation process according to one embodiment of the present invention;

Figure 11A shows a group of triangles being assigned to respective side view images; and

Figure 11B illustrates that a patch is growing with every newly added triangle.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Notation and Nomenclature

In the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will become obvious to those skilled in the art that the present invention may be practiced without these specific details. In other instances, well known methods, procedures, components, and circuitry have not been described in detail to avoid unnecessarily obscuring aspects of the present invention.

The detailed description of the present invention in the following are presented largely in terms of procedures, steps, logic blocks, processing, and other symbolic representations that resemble of data processing in computing devices. These process descriptions and representations are the means used by those experienced or skilled in the art to most effectively convey the substance of their work to others skilled in the art. The method along with the system and the computer readable medium to be described in detail below is a self-consistent sequence of processes or steps leading to a desired result. These steps or processes are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities may take the form of electrical signals capable of being stored,

transferred, combined, compared, displayed and otherwise manipulated in a computer system or electronic computing devices. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, operations, messages, terms, numbers, or the like. It should be borne in mind that all of these similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following description, it is appreciated that throughout the present invention, discussions utilizing terms such as "processing" or "computing" or "verifying" or "comparing" or the like, refer to the actions and processes of a computing device that manipulates and transforms data represented as physical quantities within the computing device's registers and memories into other data similarly represented as physical quantities within the computing device or other electronic devices.

System Overview and Image Acquisition

Referring now to the drawings, in which like numerals refer to like parts throughout the several views. **Figure 1** demonstrates a system **100** in which the present invention may be practiced. An object **102** is typically large and may not be feasible to be placed on a turntable to be rotated while being imaged. The object may include, but may not be limited to, a nature scene, terrain, man-

made architecture and parts. To image such object, a user or operator will carry a camera or an imager and produce a sequence of images in a format of video frames or a sequence of pictures by gradually moving the imager around or relatively against the object. For example, an imager is attached to a flying vehicle if a particular area of urban terrain needs to be modeled. A sequence of images of the particular area is thus generated when the flying vehicle flies over the urban terrain.

To facilitate the description of the present invention, the object **102** is assumed to be a building (e.g. a tower in the figure). Thus a user or operator can walk around object **102** to produce a sequence of images providing a surrounding view of object **102**. Imager **104** may be a video camera whose focal length is, preferably, set to a fixed known position, when the surrounding imagery is generated. Imager **104** is coupled to computer system **106** that includes a frame grabber. The frame grabber digitizes each of the video frames received from imager **104** to produce a sequence of digital images C_1, C_2, ... C_N, typically in a commonly used color format, coordinates or space. One of the commonly used color spaces is the RGB color space in which each of the image color pixels is represented as a vector $C(i, j) = [R(i, j), G(i, j), B(i, j)]^T$, where (i, j) are coordinates of an image pixel C(i, j) and R, G and B are the respective three intensity images in color image C.

It is understood that the R, G, and B color image data representation is not necessarily the best color space for certain desired computations, there are many other color spaces that may be particularly useful for one purpose or another. One of those is HIS (hue, intensity, and saturation) representation that facilitates the separation of hue, intensity, and saturation from a color image. Other possible coordinates that may possess similar characteristics to HIS may include Lu*v* and La*b*. To facilitate the description of the invention, the following embodiments assume that computer system **106** receives color images in the format of the RGB space. The description makes it evident to those skilled in the art when computer system **106** receives other than the RGB format images.

Computer system **106** may be a computing system that may include, but not be limited to, a desktop computer, a laptop computer or a portable device. **Figure 2** shows a block diagram showing an exemplary internal construction of computer system **106**. As shown in **Figure 2**, computer system **106** includes a central processing unit (CPU) **122** interfaced to a data bus **120** and a device interface **124**. CPU **122** executes certain instructions to manage all devices and interfaces coupled to data bus **120** for synchronized operations and device interface **124** may be coupled to an external device such as imaging system **108** hence image data therefrom are received into a memory or storage through data bus **120**. Also interfaced to data bus **120** is a display interface **126**, network interface **128**, printer interface **130** and floppy disk drive

interface **138**. Generally, a complied and linked version of one embodiment of the present invention is loaded into storage **136** through floppy disk drive interface **138**, network interface **128**, device interface **124** or other interfaces coupled to data bus **120**.

Main memory **132** such as random access memory (RAM) is also interfaced to data bus **120** to provide CPU **122** with the instructions and access to memory storage **136** for data and other instructions. In particular, when executing stored application program instructions, such as the complied and linked version of the present invention, CPU **122** is caused to manipulate the image data to achieve desired results. ROM (read only memory) **134** is provided for storing invariant instruction sequences such as a basic input/output operation system (BIOS) for operation of keyboard **140**, display **126** and pointing device **142** if there are any.

Feature Extraction and Tracking

One of the features in the present invention is to provide an automatic mechanism that extracts and tracks only the most salient features in the image sequence, and use them to automatically generate the motion of the imager. The features used in the present invention are those that are characterized as least altered from one frame to an adjacent frame and can be most accurately located in the image, for example, salient comer-like features in each of the image frames.

To accelerate the process of feature extraction and tracking, the present invention uses a salient feature operator to detect the features only in an initial image or those images that appear to have lost some of the features being tracked. For images subsequent to the images applied with the salient feature operator, the present invention utilizes multi-resolution hierarchical feature tracking to establish features correspondence to the features detected by the salient feature operator.

According to one embodiment, the salient features to be extracted are typically those corner-like features in the images. **Figure 3** illustrates a 3D drawing **202** of an intensity image **200** that includes a white area **204** and a dark area **206**. Drawing **202** shows a raised stage **208** corresponding to white area **204** and a flat plane **210** corresponding to dark area **206**. Corner **212** is the salient feature of interest whose location change can be the most accurately determined and typically least affected from one frame to a next frame.

A salient feature detection processing is designed to detect all the salient features in an image. The salient feature detection processing is to apply a feature detection operator to an image to detect the salient features therein. According to one embodiment, the feature detection operator or feature operator $O(I)$ on an image I is a function of the Hessian matrix of a local area of the image that is based on the Laplacian operator performed on the area. Specifically, the salient feature operator $O(I)$ can be defined as:

$$I_f = O(I) = Det[H(I)] - \lambda G(I)$$

where I_f is, as a result, defined as a feature image resulting from the salient feature detection processing by O(*I*). *Det*( ) is the determinant of matrix H and λ is a controllable scaling constant and:

$$G(I) = Ixx + Iyy$$

The Hessian matrix can be further expressed as follows:

$$H(I) = \begin{bmatrix} Ixx & Ixy \\ Ixy & Iyy \end{bmatrix}$$

where x and y are the horizontal and vertical direction, respectively, and the second order derivatives:

$$I_{xx} = \frac{\partial^2 I_s}{\partial x^2} \qquad I_{xy} = \frac{\partial^2 I_s}{\partial x \partial y} \qquad I_{yy} = \frac{\partial^2 I_s}{\partial y^2}$$

and I_s is a smooth version of image *I* by performing an image convolution with a 2D Gaussian kernel that is typically 11x11 to 15 x 15 pixels in size.

One of the unique features of the salient feature operator described herein is the ability of emphasizing only the corner-like regions, such as 212, while suppressing edge or homogeneous regions, such as **214**, **208** and **210** in **Figure 3**. After image *I* is processed by the salient feature operator, the local maximums of salient image I_f are then extracted which correspond to the salient features. Typically, image *I* is an intensity image that may be an intensity component in the HIS color space or a luminance component derived from the original color image.

Generally each of the salient features is presented as a template, such as a 11-by-11 or 13-by-13 image template. The characteristics or attributes of a salient feature template may comprise the location of the feature in the image, color information and strength thereof. The location indicates where the detected salient feature or the template is located within the image, commonly expressed in coordinates (i, j). The color information may carry color information of the template centered at (i, j). The strength may include information on how strongly the salient feature is extracted or computed as $I_f(i, j)$.

In operation, there are N color images sequentially received from an imager. As each color image is received, it is first transformed to a color space in which the luminance or intensity component may be separated from the chrominance components. As understood by those skilled in the art, the color image conversation is only needed when the original color image is presented in a format that is not suitable for the feature extraction process. For example, many color images are in the RGB color space and therefore may be preferably transformed to a color space in which the luminance component may be consolidated into an image. The above feature operator is then applied to the luminance component to produce a plurality of the salient features that preferably are indexed and kept in a table as a plurality of templates. Each of the templates may record the characteristics or attributes of each feature.

By the time the N color images are processed, there shall be N corresponding feature tables, each comprising a plurality of the salient features. The tables can then be organized as a map, referred to herein as a features tracking map, that can be used to detect how each of the features is moving from one image frame to another.

In a preferred embodiment, a multi-resolution hierarchical feature structure is used to extract the features for tracking. To be specific, **Figure 4A** shows two consecutive images **402** and **404** are successively received from imager **104**. After the salient feature operator is applied to image **402**, it is assumed that one feature **406** is detected and the characteristics thereof are recorded. When second image **404** comes in, a multi-resolution hierarchical image pyramid from the image is generated.

Figure 4B shows an exemplary multi-resolution hierarchical feature structure **408** for extracting feature **406** in image **404**. There are a number of image layers **410** (e.g. L layers) in image structure **408**. Each of the image layers **410** is successively generated from the original image **404** by a decimation process around the feature location. For example, layer **410-L** is generated by decimating layer **410-(L-1)**. The decimation factor is typically a constant, preferably equal to 2. Given the characteristics of the feature found in image **402** and knowing that two image **402** and **404** are two successive images, the feature and its location **405** in image **404** shall not alter drastically. Therefore an approximate search area for the feature can be defined in

the second image and centered at the original location of the feature. More specifically, if feature **406** is located at coordinates (152, 234) in image **402**, the window to search for the same feature may be defined as a square centered at (152, 234) in image **404**.

As the window size is predefined but the motion of the imager is unknown, there can be situations in which the feature may fall out of the predefined search window, resulting in the loss of the feature. One intuitive approach is to enlarge the search window so that the feature can be detected within the window. However, as the window size increases, the processing time is quadraticly proportionally increased. With tens or hundreds of features to be extracted in one image, the feature extraction and tracking process could become computationally very expensive.

Multi-resolution hierarchical feature structure **408** shows that a sought feature can be extracted even if it happens to fall out of a predefined search window without increasing the processing time. As the number of layers **410** is increased upward, the resolution of each of layers **410** decreases. In other words, when the size of the search window remains the same, the search area is essentially enlarged. As shown in the figure, search window **412** covers relatively a larger area in layer **410**-L than in layer **410**-(L-1). In operation, layer **410**-L is first used to find an approximated location of the feature within search window **412**. One of the available methods for finding the location of the corresponding function in the consecutive images is to use a

template matching process. The template is defined as typically a square image region (11-by-11 to 15-by-15) centered at the location of the original feature extracted by the salient feature operator. Then the corresponding subpixel accurate location of the match can be found at that position where the normalized cross-correlation of the two corresponding images regions is the largest (ideally "1" for a complete match). Layer **410**-(L-1) is then used to refine the approximated location of the feature within the closest area in the same window size and finally layer **410** is used to precisely determine the exact location (x, y) of the feature. It can be appreciated that the use of the feature structure has many advantages over prior art feature extraction approaches. In essence, an effectively larger representation of the feature template can be achieved, which makes it possible to track a feature effectively and precisely and is directly suitable to the hierarchical tracking mechanism.

Generally there are K salient features in an image and K can be in a range of 10 ~ 1000. Hence there are K feature structures like the one in **Figure 4B**. **Figure 4C** shows K feature structures **420** from a single image, each of feature structures **420** is for one feature. As a result of the feature extraction, a set of attributes F(...) describing each of the K features are produced and may comprise information of the location, strength and color of the feature.

With N image frames and K sets of the attributes $F_i(...)$, i = 1, 2, ... K, **Figure 4D** shows what is called herein a "features tracking map",

or simply, a features map that illustrates collectively all the features found for N images and is used for tracking the features so as to estimate the motion of the imager. In addition, **Figure 4E** shows a flowchart of the feature extraction process. Both of **Figures 4D-4E** are described conjointly to fully understand the feature detection and tracking process in the present invention.

At **452**, color images are successively received from the imager. A dominant component, preferably the luminance or intensity component is extracted from the color images at **454**. In one embodiment, the color images are simply transformed to another color space that provides a separate luminance component. At **456**, the process looks up, for example, a memory area, for any features or feature templates stored there. If there are sufficient number of feature templates in the memory area, that means that the process needs to proceed with feature tracking in the next image, otherwise, the process needs to check if new features must be extracted at **458**. In operation, the first received image always invokes the feature extraction operation with the salient feature operator as there are no stored features or feature templates to perform the feature tracking process. So the process now goes to **460**.

At **460**, the feature extraction process generates K features in the received image (e.g. frame #1). As illustrated in **Figure 4D**, there are K features in the received image frame #1. Preferably, the

attributes of the K features, as feature templates, are stored in a memory space for subsequent feature extraction process.

When a next image comes in at **462**, the process goes to **464** to generate the multiple-resolution hierarchical image pyramid preferably having the newly arrived image as the base. As described above and shown in **Figure 4C**, the tracking process searches for locations in the image pyramid which demonstrate most similarity to the respective layers of the feature templates stored in the feature structures. With each of the K multi-resolution feature structures, K or less corresponding features are localized from each corresponding layer in the image pyramid at **466** and the K feature locations are then collected and appended to the features map for frame 2. Similarly, for the next n1 frames, the process goes to **462** via **456** repeatedly to extract K features from each of the n1 frames.

It may be observed that, as images are generated, the imager may have been moved around the object considerably with respect to the initial position from which the first image is captured. Some of the K features may not necessarily be found in those late generated images. Because of the perspective changes and motion of the imager, those features may be either out of the view or completely changed so that they could be no longer tracked. For example, a corner of a roof of a house may be out of the view or lose its salient feature when viewed from a particular perspective. Therefore, the representation **430** of the

K features for n1 images in **Figure 4D** shows the dropping of the number of the features.

As one of the features in the present invention, the generation of features is invoked when the number of features drops exceeds a predefined threshold (T). At **456**, when it is found that a certain number of the features can not be found in an incoming image, the process goes to **458** to determine if it is necessary to extract new features to make up the K features. As described above, when the number of the features drops due to a perspective change or occlusion, new features may have to be extracted and added to maintain sufficient amount of features to be tracked in an image. The process restarts the feature detection at **460**, namely applying the salient feature operator to the image to generate a set of salient features to make up for those that have been lost. The process is shown, as an example, to restart the feature detection at frame n1 in **Figure 4D**.

If too many features are dropped despite recent feature extraction, it means that the camera has just moved considerably, which may have caused abrupt perspective changes hence reduce similarity between two adjacent image frames. In this case, the process makes an attempt to reduce inter-frame decimation by what is called "back tracking" and load a preceding image frame successively at **471** until a sufficient number of correspondence is recovered.

In one embodiment, the imager produces 30 frames of image per second. Usually, a number of consecutive images possess high correlation between each other and provide mostly redundant information as to how the features are moved from one frame to another. Therefore to eliminate redundant input data, the incoming images are sampled at a predefined rate that can be an integer starting from 1. For example, when the predefined rate is set to 10, starting with the first image, every tenth image is used as input for determining the camera motion or a 3D model of the object. **Figure 4F** shows a series of images **480** are receiving from the imager. Generally, images at frame L-th, 1L-th, 2L-th, ... are actually used. When an image **482** comes in and is determined that a certain number of features are disappeared despite a recent feature extraction, the feature tracking process in **Figure 4E** performs a back tracking at **471** before applying the salient feature operator to image **482** to generate additional feature templates at **460**. Specifically, skipped images before image **482** may be backtracked to determine exactly from which image the features are actually disappeared. For example, an immediate preceding image **484** or a middle preceding image **483** is now retrieved for feature tracking. If the features in sought are still not found in image **483** or **484**, the process goes repeatedly from **470** to **462** and back to **471** via **456** and **458** to sequentially retrieve images for feature tracking till an image frame is found between **484** and **485** with sufficient number of feature correspondence. The advantage of the backtracking provides the benefit of automatic determination of the necessary lowest frame sub-

sampling rate at which the sufficient number of feature correspondences can still be maintained.

However, if no sufficient number of feature correspondence can be established by frame decimation at **471**, the tracking process has to proceed to **472** to find an alternative that is described below.

As indicated in **Figure 4E** and also shown in **Figure 4D**, the feature templates to be matched with consecutive images remain as the original set in tracking the features in subsequent images and do not change from one frame to another. Typically establishing feature correspondence between consecutive image frames can be accomplished by two ways. One is to achieve this in directly consecutive image pairs, the other one is by fixing the first frame as reference and finding the corresponding locations in all other frames with respect to this reference frame. In one embodiment, the second approach is used since it minimizes possible bias or drifts in finding the accurate feature locations, as opposed to the first approach where significant drifts can be accumulated over several image frames. However, the second approach permits only short-lived feature persistence over a few frames as the scene viewed by the camera undergoes large changes of view as the camera covers large displacements, which ultimately causes the tracking process proceed to **472**.

In order to maintain feature tracking over many of subsequent frames using the second approach, a feature template update

mechanism is incorporated in **474**. As shown in **Figure 4G**, if no corresponding feature locations can be found in the most recent frame **492** with respect to the features in the reference frame **490**, the templates of the lost features are replaced by the ones located in the most recent frame **492** in which they have been successfully tracked, i.e. at **494**. The template update at **474** of **Figure 4E** provides the benefits that features can be successfully tracked even if they may have had a significant perspective view change by minimizing accumulative drift typical for the first approach.

Understandably, the feature regeneration is invoked after every certain number of frames. **Figure 4D** shows, respectively, feature sets **432 - 436** for images at frame number n1, n2, n3, n4, n5 and n6. The frame number n1, n2, n3, n4, n5 and n6 may not necessarily have an identical number of frames in between. As the imager further moves and generates more images, some of the features may reappear in some of the subsequent images, are shown as **438-440**, and may be reused depending on the implementation preference. At **470**, the process ensures that all the frames are processed and features thereof are obtained. As a result, a features map, as an example in **Figure 4D**, is obtained.

Estimation of Camera Motion

Estimation of camera motion as disclosed herein is an automatic process to detect from a sequence of images the actual motion parameters (translation and rotation) of the camera or imager

that has traveled to produce the sequence of images. The estimation of the camera motion has many applications, such as to combine computer graphics with live video footage, also known as match movie application in movie production or indoor/outdoor robot navigation. Those skilled in the art will appreciate that the process described below can be used independently.

Figure 5 shows a flowchart of the camera motion estimation process **500** and should be understood with **Figures 6A-6D**. At **502**, a features map having characteristics similar to the example in **Figure 4D** is used in **Figure 6A** to describe process **500**. At **504**, features are grouped respectively. Specifically shown in **Figure 6A**, features extracted from a number of successive images are grouped into a respective feature block. For example, a group of features **430** and **432** are respectively collected as a feature block **602** and **604**. As noted in the figure, there is an overlapping **606** between feature blocks **602** and **604** and further each of the feature blocks is full, namely the size of the feature block is so chosen that no features are dropped over the number of frames enclosed by the feature block. In operation, a feature block of K features and n frames is expressed as a 2K-by-n feature matrix in the following:

$$\begin{bmatrix} x11 & x12 & \cdots & x1n \\ x21 & x22 & \cdots & x2n \\ \vdots & \vdots & \ddots & \vdots \\ y11 & y12 & \cdots & y12 \\ y21 & y22 & \cdots & y2n \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix}$$

where (*x*ij, *y*ij) is the coordinates of a feature, i is a i-th feature and j is a j-th frame. The size of the overlapping may be, for example, 10 to 30 features versus 3 to 10 frames. In other words, the first and last few columns of the above feature matrix are generally for features in the overlapping. The overlapping, as will be further described below, provides information to concatenate camera motion segments derived respectively from each feature block.

A complete camera motion comprises a number of small motion segments that are each respectively derived from one of the feature blocks. As shown in the figure, there is an overlapping between each pair of two adjacent feature blocks, such as overlapping **606** between feature blocks **610** and **612** to provide information to concatenate two motion segments from feature blocks **610** and **612**, so as to form a concatenated motion of the camera.

At **506**, a first feature block is taken for processing. At **508**, process **500** adjusts the positions of the features in the feature block with feedback information from **515**. The positions of the features as well as the detailed feedback information will be described below. Generally when the feature attributes in a feature block are provided for the first time, process **500** does not adjust the positions of the

features in the feature block but instead transfer the features directly to factorization process **510** to compute initial solutions.

The factorization process at **510** is an intermediate process that can recover shape and motion from a series of image under orthography. The detailed description is provided by Tomasi and Kanade, "Shape and Motion from Image Streams under Orthography: a Factorization Method," International Journal of Computer Vision Volume 912, 1992, pp.137-154, which is hereby incorporated by reference in its entirety. The factorization process takes as an input a feature matrix and outputs a camera rotation and object shape. As described above, a feature matrix for a feature block having a size of K-by n is a 2K-by-n matrix. The 2K-by-n matrix is then factored, under certain constraints, into a product of two matrixes **R** and **S**, where **R** is a 2K-by-3 matrix that represents the camera rotation and **S** is a 3-by-n matrix that represents shape in a coordinates system in which the object is positioned. It should be noted that the outputs are correct only when the camera model is orthographic.

In addition, from factorization process at **510**, 2D image translation information Tfq for each frame can be obtained. As shown in **Figure 6B**, Tfq represents the displacement of a scene point located at Pfq projected onto the image.

At **512**, given the focal length of the camera, the initial estimate of the average absolute distance Zo between an image and the object is computed. It should be pointed out that both of the focal length and

the average absolute distance Zo need not to be accurate and will be refined to a predefined accuracy through an iterative feedback adjustment, which is described in detail below.

At **514**, translation and shape matrix need to be refined with respect to the calculated average depth Zo with the result from the factorization process at **514**. It is noted, however, that the factorization method proposed by Tomasi and Kanade is based on an assumption that the object is orthographically projected. Therefore the results from factorization process at **510** can not be directly applicable to the determination of the camera motion because the object may not be distantly located from the camera. The orthographic projection is a restrictive and quite often unrealistic condition. In many real applications, an imager or camera often has a field of view larger than 10 ~ 20 degrees and therefore the projection of an object to an image plane has to be a perspective model. As one of the important features in the present invention, an iterative feedback mechanism is used to adjust the coordinates of the features as the input to factorization process at **510**. The underlying feedback mechanism may be better understood in accordance with **Figure 6C** which is implemented as one embodiment of the present invention.

From a camera perspective, normal perspective camera **630** means a regular video camera that is used to generate the sequence of images. To satisfy the unique conditions in which the factorization method **634** works right, normal perspective camera **630** goes through

an orthographic correction **632** of the feature locations. The outputs from factorization **634** are further refined by least square (LS) triangulation using the average depth obtained through the focal length characterizing camera **630**. The refined outputs are then fed back to correction **632** that further adjusts perspective data of normal perspective camera **630** to orthographic data until the unique conditions in which the factorization method **634** works right are closely approximated, hence resulting in correct outputs. In a particular example, the adjustment of the perspective data is done by extending the positions of the features outward from the image center as a function of their distances from the camera.

Figure 6D illustrates how a cube **650** is projected and corrected through perspective correction **632**. When cube **650** is not distantly positioned in the field of view of normal perspective camera **630**, a perspectively distorted cube **652** will be produced in an image due to the different distance of the points on cube **650** from the camera. While the unique condition for the factorization method **634** to work right is the orthographic projection, noticeably large errors will inevitably result when a regular image (e.g. cube **652**) is provided. As described above, the errors are used to adjust the image so that an adjusted image gets closer to an image obtained under the orthographic projection. The adjusted image is then provided to factorization method **634**, smaller errors will be produced, the errors are then again used to further adjust the image. The feedback processing keeps going on (e.g. 10 loops) until the adjusted image

gets very close to an image that would be otherwise produced under the orthographic projection.

Referring back to **Figure 6C**, the outputs from factorization **634** include Rfq, Tfq, Cfq and Pfq for each of the image frames. Rfq and Pfq, rotation information and scene coordinates of a feature, are corresponding elements in the rotation matrix **R** and shape matrix **S**. Tfq and Cfq are the corresponding translation and scaling factor. As shown in the figure, Pfq and the approximated focal length is used to refine the average distance Zo in **636** that is provided to the least square estimation of 3D points Pe **636**. With the refined 3D points Pe, the averaged distance Zo, the camera translation is refined to Te using the least square triangulation . It is noted therein that the rotation Rfq or the rotation matrix R are not further refined as it has been produced with the refined Te and Pe. Depending on a predefined degree of precision, all the refined values are iteratively provided as feedback signals to correction **632** so that subsequent refined values become accurate enough to derive the camera motion .

Referring now back to **Figure 5**, the decision to continue the above iterative refining process is carried out at **515**. In other words, if the back projection errors with respect to their original locations are not small enough or the number of iterations is not reached, process **500** will go back repeatedly to **508** until the back projection errors become less than predefined thresholds or the number of iterations is reached. At **516**, process **500** checks if the feature map is complete at **516**. In

other words, a feature block including the last image frame shall be processed and the refined rotation matrix **R** and shape matrix **S** thereof are derived.

As a result of the feedback mechanism described above, each feature block produces a set of accurate rotation Re and 3D points Pe. As shown in **Figure 7A**, each of the feature blocks **702** produces a camera motion segment for the corresponding frame interval, which includes the camera positions and orientations in the particular image frame..

More specifically, feature block **702-1** includes a number of image frames and each of the frames corresponds to a vertex in the camera motion segment **704-1**. It is understood that, except for the first and the last ones, each of the camera motion segments **704** has a certain number of overlaps with the neighboring ones by construction For example, given an overlapping of 3 frames, the last 3 vertices of motion segment **704-1** should coincide with the first 3 vertices of motion segment **704-2**, and the last 3 vertices of motion segment **704-2** should coincide with the first 3 vertices of motion segment **704-3**.

After obtaining all the motion segments, process **500** proceeds to stitch the segments together to form the total camera motion at **520**. **Figure 7B** shows how two motion segments **704-1** and **704-2** are stitched to form a concatenated motion segment**710**. With the known knowledge of the overlapping between the feature blocks that

produces motion segments **704-1** and **704-2**, the overlapping vertices **706** and **708** are used as constraints to determine the common motion . Since vertices **706** and **708** are from the overlapping and hence coincidental, motion segment **704-2** is rotated, translated and scaled to coincide with end vertices **706** of motion segment **704-1**, resulting in a concatenated motion segment **710** with vertices **706** and **708** coincided at **712**. With all the motion segments stitched together as described above, the whole camera motion can be obtained. **Figure 7C** shows an exemplary camera motion .

As noted from the above example, motion segment **704-2** has been rotated, translated and scaled to be stitched with motion segment **704-1**. The derived 3D points as well as the camera motion segments are placed in a common coordinate system by rotation, translation, and scale. To further minimize errors in the unified 3D points and the whole camera motion, a global nonlinear optimization **522** is employed to refine the parameters, which reduces the difference between the extracted feature locations and their corresponding backprojected 3D coordinates. This process provides the final, globally optimized rotation and translation of the camera motion and the 3D locations of the features. At **524**, the errors are examined if they are within the predefined range, otherwise the optimization and adjustment process is repeated till the errors are within the predefined range.

Depth Mapping Process

The depth mapping process in the present invention is an automatic process to generate high density surface points for a subsequent mesh model generation. Each of the surface points are represented in the scene space. As described above, the results from the camera motion process include the rotation Re, translation Te parameters, and 3D coordinates for each of the salient feature points. However, those salient feature points represent only a small portion of points that would not be sufficient by farto generate a surface wire frame or mesh model of the object. In reality, more feature points, referred to as dense points herein, may be expected, which typically include all boundaries and edge points located in high contrast areas. The number of such dense points can be in the range from 1000 to 100,000 for a regular object in the scene space. Using feature tracking techniques, one can establish feature correspondence for these dense features as well, and recover their 3D locations as it is descried more detail below.

Figure 8 shows a flowchart of the depth mapping process and should be understood with **Figures 6A-6C** and **9A-9C**. At **802**, the results of the camera motion, particularly the refined rotations **R** and translations **T**, are received. They can be used to provide constrained search and assist to determine the geometric location of the dense points by triangulation.

At **804**, a first image frame is retrieved for extracting dense points. At **806**, the number of currently tracked points is examined. If this number is below a threshold (a particular case for the first frame), the process moves to **808** and **810** to detect any available dense points. First straight line segments are extracted as they are the most accurate and stable features for generating a mesh model. Other advantages of detecting line feature include significant reduction of computation and persistent accuracy as will be appreciated in the following description.

There are a few approaches for detecting lines in an image. One of them is based on the Hough transform known to those skilled in the art. An image is transformed to the Hough domain in which dominant linear features in the image are detected. According to one embodiment, the exact lines are determined respectively from the underlying points using a point grouping and line fitting technique at subpixel level precision. Subsequently, each line segment extracted is represented by two end points at **912**. For example, **Figure 9A** illustrates that a house image **900** is detected for the line features of which line **908** is shown in image **902** and points **906** and **908** represents line **908**.

With all possible line features determined at **808**, the line detection, however, does not result in any points around non-line shaped parts, for example an ellipse **904** in **Figure 9A**. The dense points other than the line type need to be detected next at **810**. To

avoid the detection in the area containing the lines, a mask is preferably placed around the line area as shown **916** in image **912**. At **810**, an operator is applied to detect those non-line type dense points. The operator may be one of those edge detectors that essentially detects all the significant points located around high contrast areas. Ellipse **904** in **Figure 9A** is detected as edge points **914** in image **912** when the edge detector is applied, wherein a dotted block **916** shows an exemplary mask around line **910** that has been detected.

The above steps are typically applied at the feature extraction phase of dense point reconstruction. With the known camera motion, i.e. the rotation and translation thereof, the dense points for the rest of the images can now be tracked along respective epipolar lines rather than in unconstrained large areas, which is referred to herein as "constrained tracking".

Referring to **Figure 9B**, there is shown an object point P being projected on two adjacent image planes **920** and **922** at (x1, y1) and (x2, y2) respectively. The camera projection centers (focal points) are at **925** and **927**. Together with the object point P, the two points **925** and **927** forms an epipolar plane **928** that intersects both image planes **920** and **922**. The intersecting lines **921** and **923** are the epipolar lines for image points **924** and **926**.

If the process in **Figure 8** has detected a dense point characterizing the projected point **924** at (x1, y1) in image plane **920**, the coordinates (x2, y2) of the projected point **926** in image plane

922must lie on the epipolar line. In other words, once the epipolar line is determined for a projected point in the first image, the 3D point must be projected onto this epipolar line in the second consecutive image as it is illustrated in **Figure 9B**. The problem of tracking lines segments between frames is reduced to tracking sparsely subsampled points of the line segments and then robustly detecting line segments on the basis of tracked points in the second image.

Referring now back to **Figure 8**, after the dense points including sparsely subsampled points of all the line features are obtained, a next image is obtained at **812** for tracking the dense points along with respective epipolar lines at **814**. The match along an epipolar line can be found by performing sub-pixel accurate template matching using, for example, normalized correlation computation. At **816** and **818**, the 3D coordinates of the dense line points and those non-line points extracted above are respectively reconstructed using LS triangulation.

At **820**, the process needs to checks if all the images have been processed for the dense points. If there are still one or more images that shall be proceeded, the process goes back to **806**. As there are sufficient detected line and non-line feature points, the process will proceed with **812** to **820**.

As a result of the process in **Figure** 8, a 3D cloud of dense points and a large number of 3D line segments are computed. The next operation is to generate a mesh model of the object based on the dense points obtained from each of those images.

Meshing and Texture Mapping

Before creating a fully textured 3D model of an object, a description of the surface of the object is needed. Typically, a mesh model of the object is a desired description of the surface, as it provides the information how each localized area is oriented and positioned in a scene space so that corresponding texture information may be applied thereto to generate subsequently a fully textured 3D model. Further, a mesh model may be used as a basis to create a display or reproduction of the real world object and generate other displays such as "morphs", fantasy or special effects.

The generation of a 3D mesh model is a process that generates a mesh model of an object by dividing its 3D surface into a set of small triangular (or quadrilateral) elements. The input to the process is a list of the dense points obtained from the depth mapping described above, and the output of the process is a list of facets of the convex hull of the points with vertices defined at the point locations. The process is based on computing the 3D Delaunay triangulation that is a well known methodology to compute a triangular mesh based on a set of surface dense points.

Figure 9C shows a flowchart of generating a mesh model based on the 3D Delaunay triangulation. Since the 3D Delaunay triangle is defined only on 3D points, each line segment needs to be subsampled into points.. Letting the subsampling density sufficiently

high, the triangulation will likely have a set of edges connecting the points of the subsampled line segments with each other, which coincide with the original underlying line segment as preferred. At **942**, the dense points including the line and non-line type feature points are obtained from the above depth mapping process. At **944**, those line type feature points are identified and sub-sampled into sparse feature points, before coming to **948** to compute the 3D Delaunay triangulation. At **948**, the 3D Delaunay triangles are computed based on the individual feature points.

Generally, the triangle facets computed by the Delaunay triangulation that are based on the supplied dense points may include invalid triangles that do not correspond to true physical surface of the object in a scene in addition to unusual triangles, such as very elongated or inproportional triangles. Before applying texture information to each of these triangles, it is necessary to apply a post-processing to eliminate or merge the invalid triangles. The 3D Delaunay triangulation generates a set of tetrahedrons which occupies the convex hull of the set of dense 3D feature points. Therefore, it usually contains many triangle facets which do not correspond to true physical surfaces of the scene being observed. In order to eliminate these triangles from the 3D mesh a sequence of post-processing steps has to be performed on the 3D triangular mesh which is based on using various geometric constraints.

According to one embodiment, three steps in the post-processing are applied. At **950**, constraints based upon the visibility of image features are used to eliminate those triangles that can not be valid surface. Specifically, each feature point visible in the input sequence has to be also visible in the generated 3D mesh from the same viewpoints. No triangle facet generated in the mesh is allowed to occlude the point from any of those camera viewpoints where the point was visible in that image. If such a visible point is occluded by any triangles, the triangles have to be removed from the mesh.

.At **952**, the area and edge length of the triangles are analyzed and determined if the triangle shall be merged with a neighboring triangle. Given the dense 3D point cloud obtained by the dense reconstruction process, triangles with large edge-length or large area usually do not correspond to true physical surfaces, therefore all these triangles are eliminated from the mesh. In addition, very small or narrow triangles can be sources of numerical errors both in the visibility constraint and in texture mapping therefore they are also eliminated.

And finally, texture consistency of the triangles across several views is applied at **954** to check the consistency of the texture. Specifically if a triangle facet of the surface mesh corresponds to a true physical surface patch, the texture defined by projecting the images on the triangle from all the views where the triangle is visible has to be consistent. Conversely, if a triangle does not correspond to a true physical surface the image projections

from the visible views may define different texture maps on that triangle. For instance, if a triangle facet has been defined through a point on a roof of a house, a point on the ground and a point on a tree, then one image frame may project the side wall of the house as texture on the triangle, whereas an other view may project the sky. This inconsistency indicates that this triangle can not be a true physical surface therefore it should not be included in the 3D mesh. The check of texture inconsistency can be performed by e.g. computing the normalized cross-correlation of the corresponding texture maps.

As a result of applying these constraints above to the 3D Delaunay mesh, the final surface mesh can be obtained which corresponds to the preferably true physical surface estimated from the given sequences of image frames.

It should be pointed out the three steps at **950**, **952** and **954** are exemplary steps to post-process the triangles computed from the Delaunay triangulation. There can be other approaches know to those skilled in the art to further refine the mesh model to a predefined degree of refinement.

As the result of the mesh process, a mesh model of triangular mesh is obtained. In operation, the next step at **956** is to add texture patterns to the mesh model to enhance the realism of the model. The process itself is called texture mapping, an image synthesis technique in which a 2D image, also known as a texture image, is mapped onto a

surface of a 3D mesh model. Although there have been several texture mapping techniques around, one of the important features in the texture mapping disclosed herein is the generation of patches with contiguous texture mapping without user intervention. Another important feature is a mechanism provided to export the patches in a commonly used image file that can be subsequently modified with an image processing application.

Figure 10A shows a process flowchart of applying the texture patterns to the mesh model. At **1002**, a mesh model is received and preferably described in triangles. It should be pointed out that those skilled in that art will appreciate that the texturing process disclosed herein works with a mesh model of other shapes of polygons. Although in the preferred mode, these polygons are triangular, in other modes, they may be rectangular, hexagonal or the like. When using polygons of order greater than three, special steps may be required to ensure that all of the vertices lie within a common plane. Essentially, higher order polygons can be reduced to triangles (polygons of order 3) for convenience in processing. To facilitate the description of the texturing process, the mesh model is assumed to be of triangles and those skilled in the art will appreciate that the description herein is equally applied to a mesh model with polygons of order greater than three.

Preferably, the mesh model may be modified at **1004**, depending on a desired resolution or a degree of refinement. The

approach used at **1004** may include a decimation process which according to a set of rules reduces the number of triangles to facilitate an efficient and effective texture mapping process to be followed. The rules may include a normal comparison between two or more neighboring triangles. If a normal of one triangle is similar to a neighboring triangle within a predefined degree of refinement, the corresponding triangle may be merged together with the neighboring triangle. In addition, a user may subdivide the mesh model into one or more logic parts for texture mapping at **1004** either within the current process or using a commercially available tool, such as 3D Studio MAX in which the mesh model can be displayed and interacted with.

At **1006**, each of the triangles, based on the normal thereof, is assigned to a side view image Ci. To be specific, **Figure 11A** shows a group of triangles being assigned to respective side view images. As described above, a surrounding view of the object has been captured in a number of side view images C1, C2, ... CN, each taken at a known position relative to the object. Based on the normal of each of the triangles and the known angle of each of the side view images, each of the triangles can be respectively assigned to one of the side view images C1, C2, ... CN. A visibility test is applied for every triangle and a side view in order to ensure that the triangle is visible from the chosen side view. If the triangle is not visible from the chosen side view, an alternative side needs to be selected.

Because the triangles, even next to each other, are quite inhomogeneous, it is not uncommon that two neighboring triangles are assigned to two different side view images, which result in texture discontinuity between them if no further process is applied. For example, triangle **1102** is assigned to image C1, the neighboring triangle **1104** may be assigned to image C4 that is taken from a quite different view angle from image C1. At **1008**, each triangle assigned to a side view image is mapped to/with the side view image for texturing, namely with the patch corresponding to the portion of texture information for the triangle. At **1010**, a local blending process is applied to smooth those texture discontinuities. Additional information of process **1006**, **1008** and **1010** is provided by W. Niem, et al "Mapping Texture From Multiple Camera Views Onto 3D-Object Models for Computer Animation", the proceedings of the International Workshop on Stereoscopic and Three Dimensional Imaging, September 6 - 8, 1995, Santorini, Greece.

As one of the important features in the present invention, the generation of exportable patches is introduced herein. A patch is a collection of triangles of the mesh with the property that every triangle in the patch shares at least one edge with some other triangle in the same patch. In addition, all patches have the properties that the union of all the patches contains all the triangles of the mesh, and that no two patches contain the same triangle. Exporting such patches in image files makes it possible for a user to alter or modify the texture mapping for a particular patch in a

desirable way. For example, a 3D modeling system, typically, is not designed to model the bottom of a 3D object that is often assumed black or a color extended from what is on the bottom portion of the object. Consequently, the final 3D model loses its realism when its bottom is caused to be displayed. In other situations, users desire to remove certain reflections (e.g. speculums) caused by non-uniform illumination. With the generated textured patches, the user may use an image or graphics application, such as PhotoShop 5.0 from Adobe Systems, Inc. in San Jose, California, to manually alter or modify the textured patches. The editability of the texture mapping, and therefore the usability thereof increases tremendously if the mapping is performed in a fashion which maps neighboring triangles of a mesh to neighboring triangles in a texture image.

At **1012**, therefore, a procedure is provided to generate one or more patches, alternatively, it is to subdivide the mesh into a patch or patches. The detail of **1012** is provided in **Figure 10B**. At **1020** of **Figure 10B**, an empty patch is created (i.e. a memory space is initiated) and indexed. At **1022**, one of the triangles in the mesh model is chosen as a seed triangle. The seed triangle may be chosen randomly from the triangles that are not included in a patch yet or from a group of local triangles that demonstrate a similar normal. At **1024**, neighboring triangles to the seed triangle are sequentially checked if they have been tested for suitability to be included in the patch that is to be described below. If the

neighboring triangles are all tested, that means the patch is finished. Otherwise, the triangles are further respectively tested at **1026** to see if any of the triangles can be added to the patch.

To be specific, **Figure 11B** illustrates that a patch is growing with every newly added triangle. For example, triangle **1110** is a seed triangle that begins the patch initiated at **1020**. When a neighboring triangle **1112** has not been "tested", triangle **1112** will be tested to see if it shares at least one edge with the seed triangle. If it is not, it means that the triangle does not belong to the patch or that it may be added to the patch later in the process. As an example, neighboring triangle **1114** does not belong to the patch and will be thus discarded for the time being. If triangle **1112** shares one edge with triangle **1110**.

A mapping is created therefore at **1028** of **Figure 10B**. It should be emphasized that the particular mapping in the current embodiment is based on the orthographic projection from the 3D model to the texture image. For a particular patch, the projection is along the direction of the face normal of the seed triangle. Alternatively, the perspective projection may be used or any other suitable projections may be used.

At **1030**, the accepted triangle is further tested to see if it intersects the patch. If it does, the triangle is labeled "tested", and the process goes to **1024** to test another triangle. If the triangle does not intersect the patch, it is now added to the patch at **1034** so

that the patch grows one triangle bigger. The patch generation process permits to generate multiple patches. At **1036**, it checks if the entire mesh model has been processed, namely expressed now in a number of patches. If there are still some triangles that have not been put into a patch, then the process goes to **1020** to generate a new patch.

It can be appreciated that the patch generation process in **Figure 10B** can be implemented by a recursive programming and subsequently produces a number of mutually exclusive patches, each comprising a plurality of triangles that share at least one edge with other triangles in the patch.

At **1014**, the process is to create texture image or images. These are the images that store the actual texture. The creation of this image requires that the textures stored for every triangle are projected into the image. In the current embodiment, we accelerate the process by using graphics accelerator architecture. If such architecture is not available, the architecture is emulated by software.

As a result, the shape of patch **1118** is formed and the textured triangles therein provide a textured patch that can be saved or exported at **1016** in a commonly used image format, such as TIFF (Tag Image File Format) or JPEG (Joint Photographic Experts Group), that can be opened by an image processing application such as PhotoShop. A user can repaint or modify any

portion of the textured patch using the PhotoShop that provides sufficient graphic user interface to modify the patch at pixel level.

The process described above shows a method for creating contiguous texture patches. Rather than mapping texture to each of the triangles of the mesh model, the process chooses to map the texture from every triangle into a respective portion of the texture image. As another important features, the texture mapping process described herein can be implemented to take advantage of the graphics accelerator architecture commonly in most computer systems. Redirecting the graphics accelerator to draw into a buffer in memory rather than the buffer for the monitor can yield a much more efficient mapping of the textures.

The advantages of the invention are numerous. Several advantages that embodiments of the invention may include are as follows. One of the advantages is an economical and efficient 3D modeling system that is low in cost and easy to operate, virtually anywhere within minutes. The modeling system employing the present invention can be used and operated by an ordinary skilled person to generate fully-textured models of 3D objects within a limited time for many applications including Internet commerce and product designs. Another advantage is the MAE scheme that encodes all mask images to make the space carving process nearly independent of the size of images. Still another advantage is the process of generating a mesh model using neighborhood

configuration that produces only valid triangles. Still another advantage is the texture mapping process that provides a mechanism to generate exportable patches comprising triangles that can be provided contiguous texture mapping without user intervention. Yet another advantage is the possible implementation of the texture mapping processing on graphics accelerator architecture to redirecte the graphics accelerator to draw into a buffer in memory rather than the buffer for a monitor, yielding a much more efficient mapping of the textures As a result of the texture mapping, a fully-texture 3D model of an object is created. The advantages of the invention are numerous. Several advantages that embodiments of the invention may include are as follows. One of the advantages is the use of efficient feature extraction and tracking mechanisms to track salient features in a sequence of images. The feature extraction mechanism uses a salient feature operator to accurately and unbiasedly locate salient features based on a 3D representation of the image intensity/color. Another advantage is the use of a factorization approach under orthography. A feedback system emulates the orthographic camera model by iteratively "correcting" the perspective camera model so that the factorization approach provides practical and accurate solutions. Still another advantage is the texture mapping process that provides a mechanism to generate exportable patches comprising triangles that can be provided contiguous texture mapping without user intervention.

The present invention has been described in sufficient detail with a certain degree of particularity. It is understood to those skilled in the art that the present disclosure of embodiments has been made by way of examples only and that numerous changes in the arrangement and combination of parts as well as steps may be resorted without departing from the spirit and scope of the invention as claimed. Accordingly, the scope of the present invention is defined by the appended claims rather than the forgoing description of embodiments.

CLAIMS

We claim:

1. A method for generating a fully-textured 3D model of an object, said method comprising:

receiving a sequence of images from an imager that is in motion relative to said object;

extracting features that are least variant from one to another in said images for each of said images;

inputting groups of said extracted features, respectively, to a camera motion estimation process; each of said groups corresponding to a certain number of said images that include said features in said each of said groups; and

determining 3D locations of dense points based on respective feature tracking constraints determined by outputs from said camera motion estimation process.

2. The method as recited in claim 1 further comprising:

generating a mesh model comprising triangles based on said extracted dense points; and

mapping said triangles respectively with texture information from said sequence of images to generate said fully-textured 3D model of said object.

3. The method as recited in claim 2, wherein said extracting features comprises:

applying a feature operator to a first image of said images to detect said features; and

tracking said features in images subsequent to said first image in said images.

4. The method as recited in claim 3, wherein said feature operator, when applied to said first image, emphases salient corner-like regions while suppressing edge-like and homogeneous regions.

5. The method as recited in claim 4, wherein said feature operator is based on a function of the Hessian matrix comprising Laplacian operator and performing on an area of a smoothed version of said first image.

6. The method as recited in claim 2, wherein said camera motion estimation process comprises a factorization process working under a orthographic condition.

7. The method as recited in claim 6, wherein said inputting comprises:

refining said outputs to approximate said orthographic condition by correcting recursively locations of said features with respect to said outputs.

8. The method as recited in claim 7, wherein said feature tracking constraints are respective epipolar lines, each for one of said dense points.

9. The method as recited in claim 8, wherein said extracting dense points comprises:

detecting said dense points in a first image of said images;

determining said respective epipolar lines, each for one of said dense points; and

tracking said dense points respectively along said respective epipolar lines in subsequent images to said first image in said images.

10. The method as recited in claim 9, wherein said detecting said dense points further comprises:

detecting first-type points representing line features in said first image;

detecting second-type points representing other than said line features in said first image; and

wherein said dense comprises said first-type points and said second-type points.

11. The method as recited in claim 2 further comprising:

generating patches based on said triangles, each of said patches being a collection of some of said triangles of said mesh model; wherein each of said triangles in said patch shares at least one edge with other triangles in said patch; and

wherein a union of all said patches contains all said triangles of said mesh model, and no two of said patches contain the same triangle therein.

12. The method as recited in claim 10, wherein said patches are exportable in a commonly used image file format so that a user can alter textured surface independently in said patches.

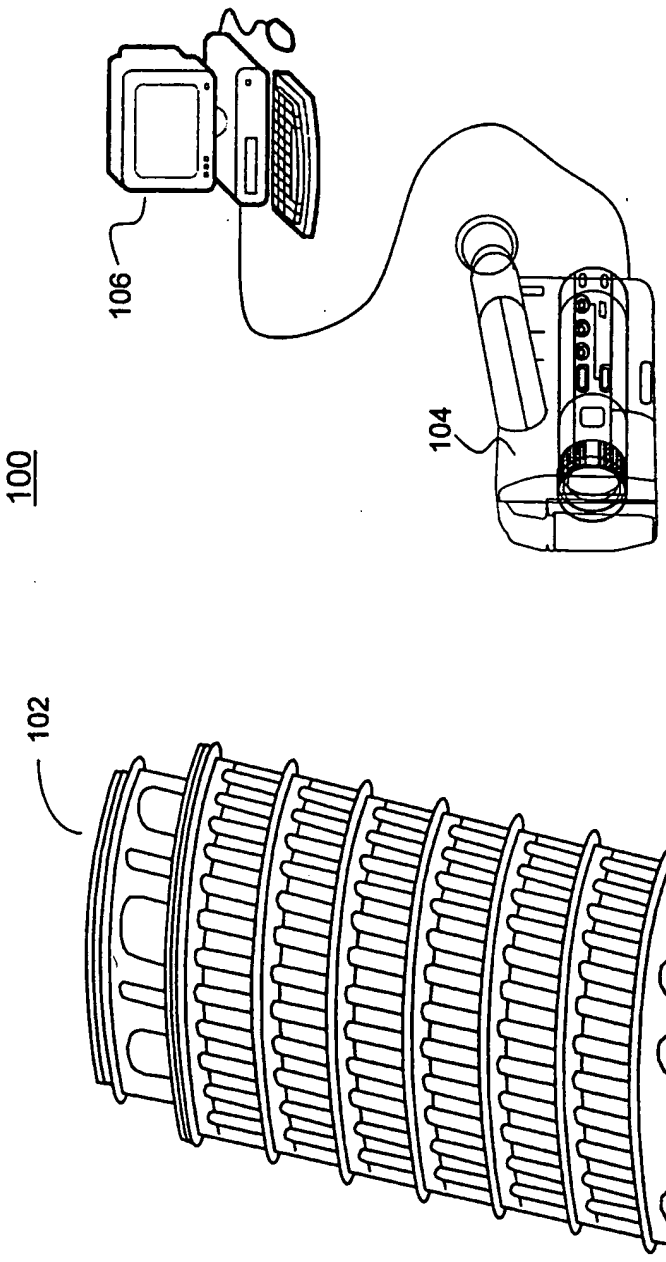


Fig. 1

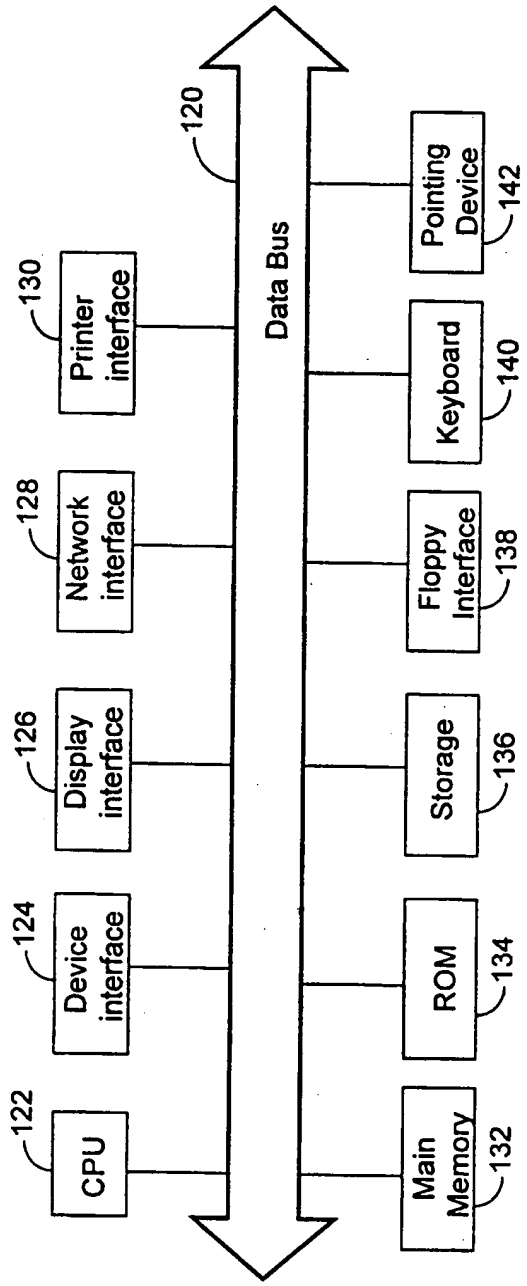

Data Bus
120
CPU
122
Device interface
124
Display interface
126
Network interface
128
Printer interface
130
Main Memory
132
ROM
134
Storage
136
Floppy Interface
138
Keyboard
140
Pointing Device
142

Fig. 2

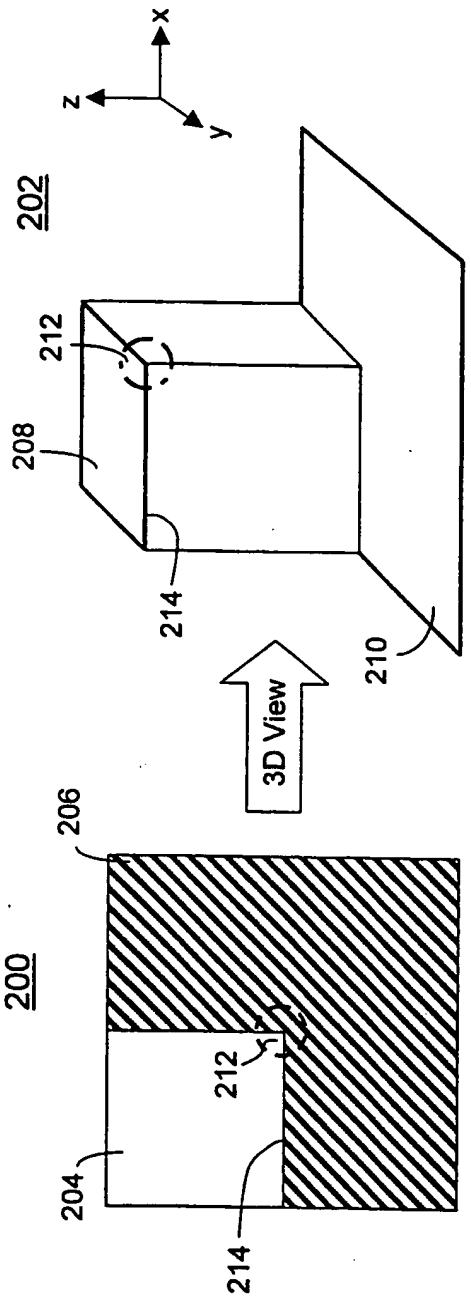


Fig.3

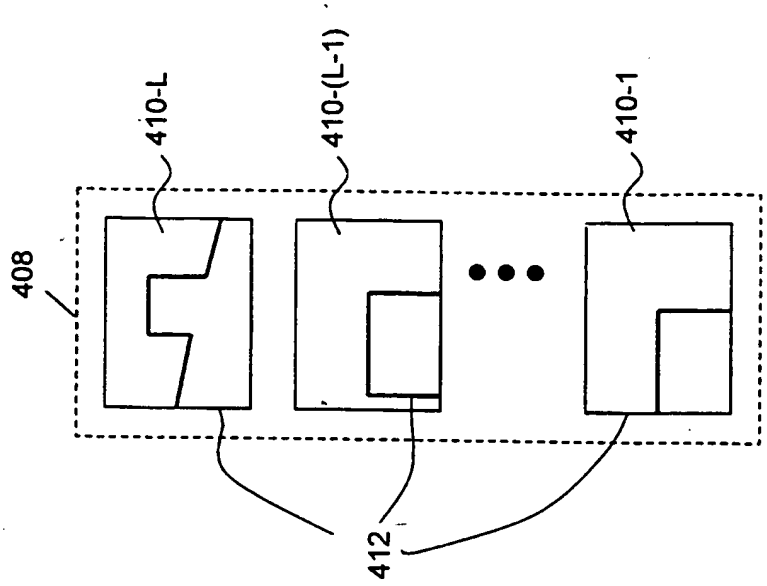


Fig.4B

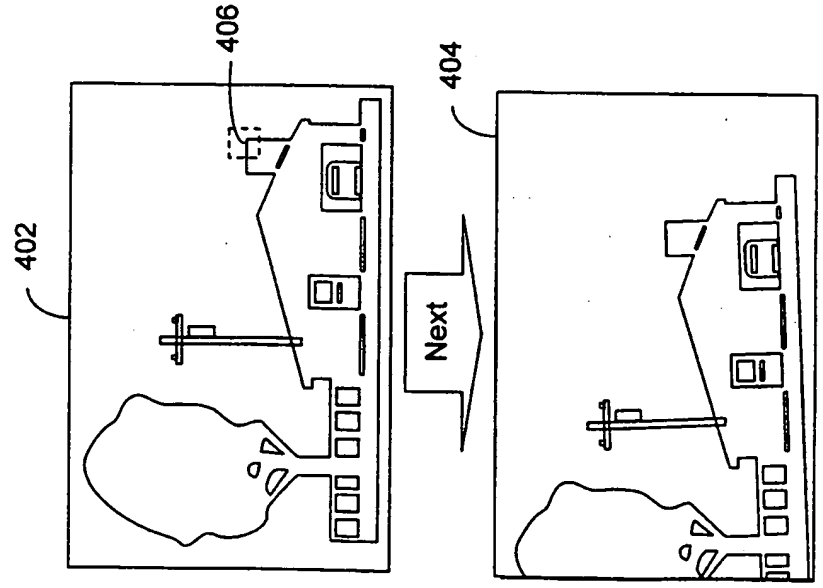
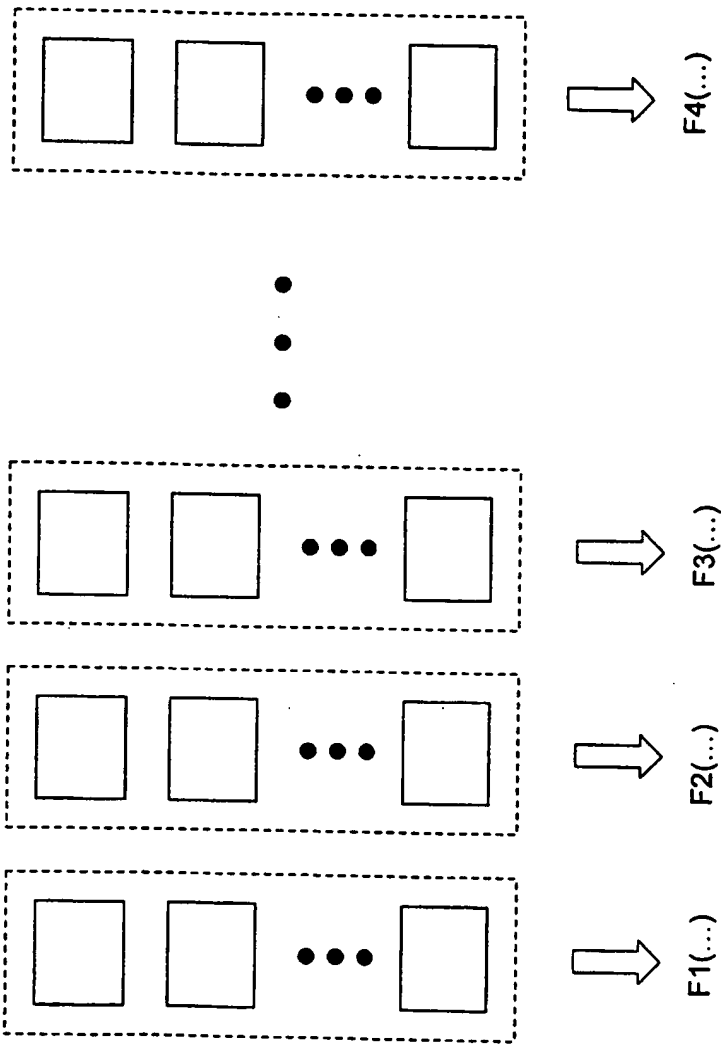


Fig.4A

Fig.4C



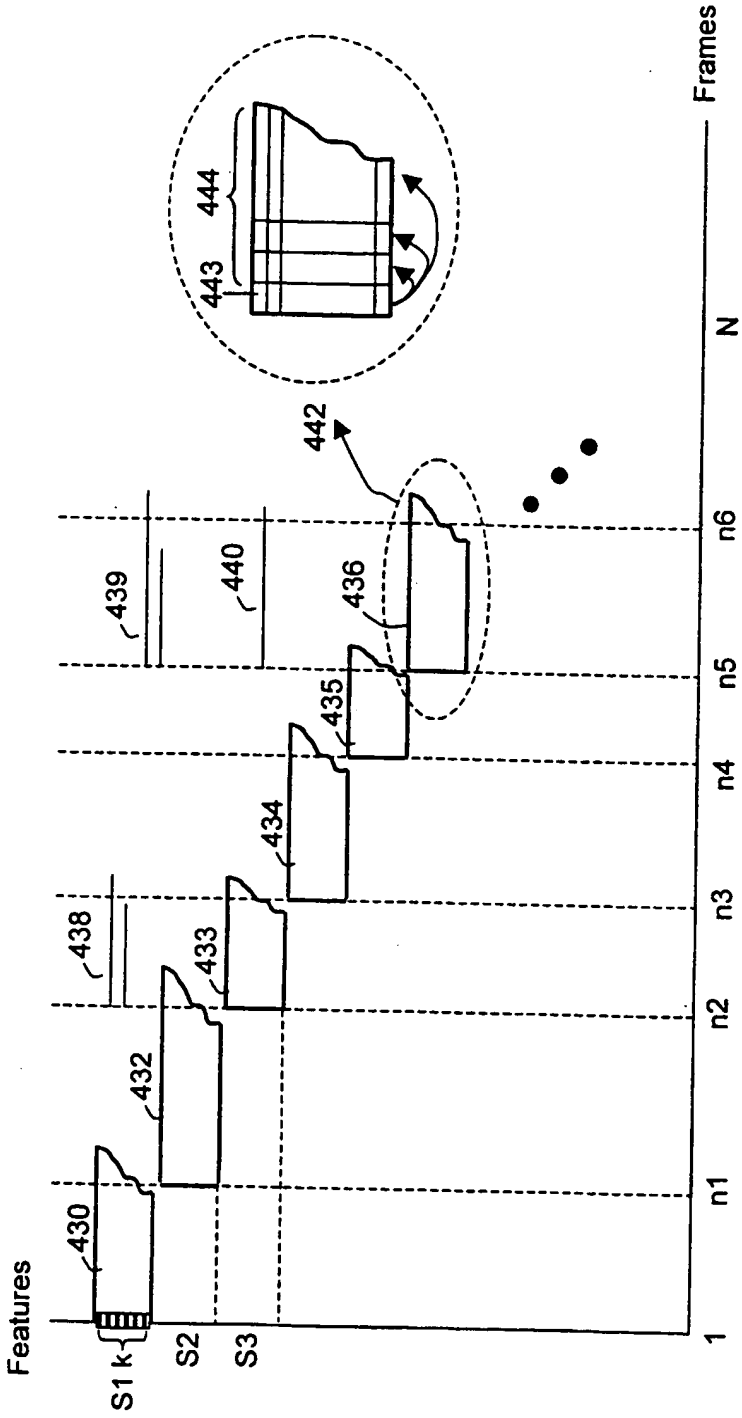

Frames
N
n6
n5
n4
n3
n2
n1
1
Features
S1 k
S2
S3
430
432
433
434
435
436
438
439
440
442
443
444

Fig.4D

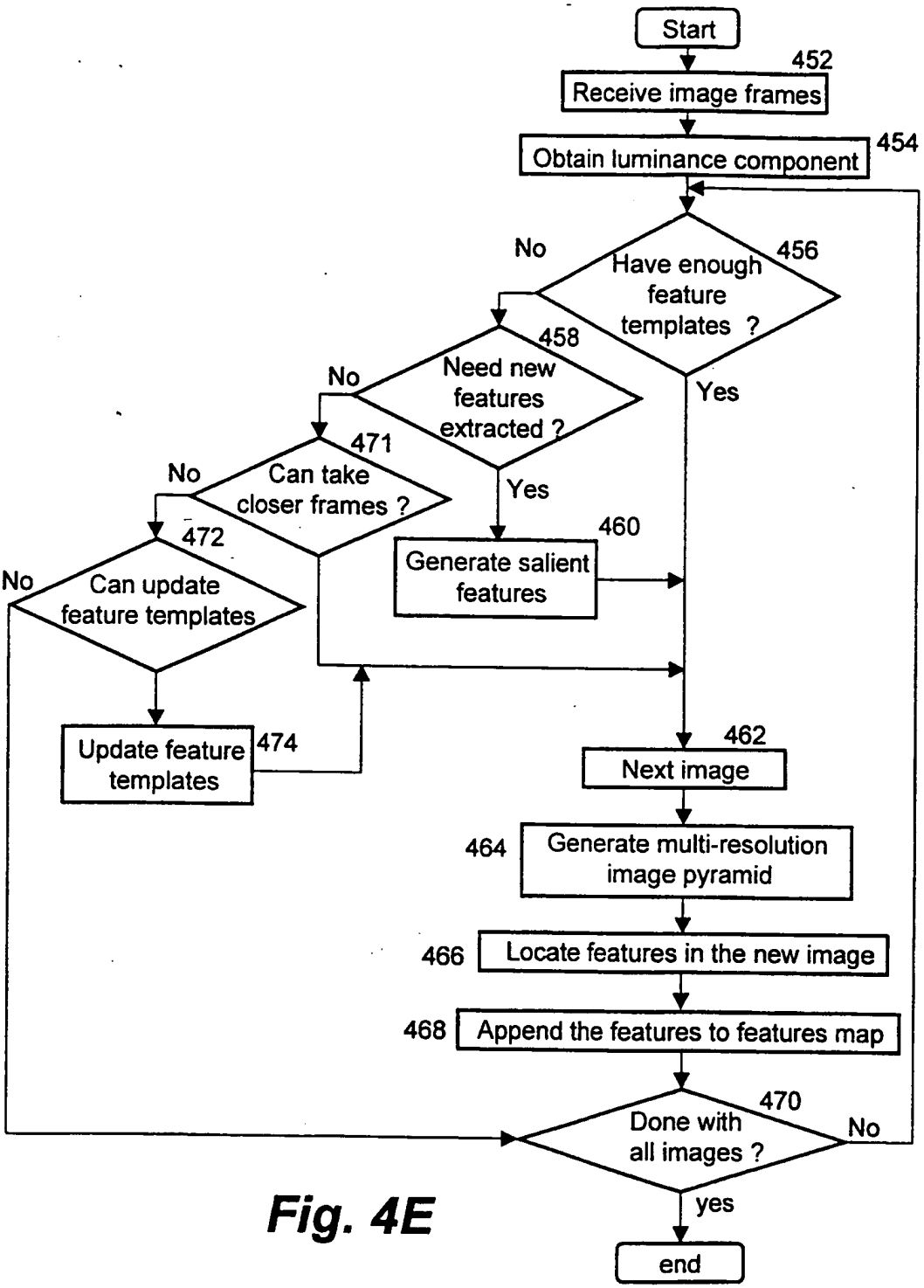

Start
452
Receive image frames
454
Obtain luminance component
No
456
Have enough feature templates ?
Yes
458
No
Need new features extracted ?
Yes
471
No
Can take closer frames ?
472
No
Can update feature templates
460
Generate salient features
474
Update feature templates
462
Next image
464
Generate multi-resolution image pyramid
466
Locate features in the new image
468
Append the features to features map
470
Done with all images ?
No
yes
end

Fig. 4E

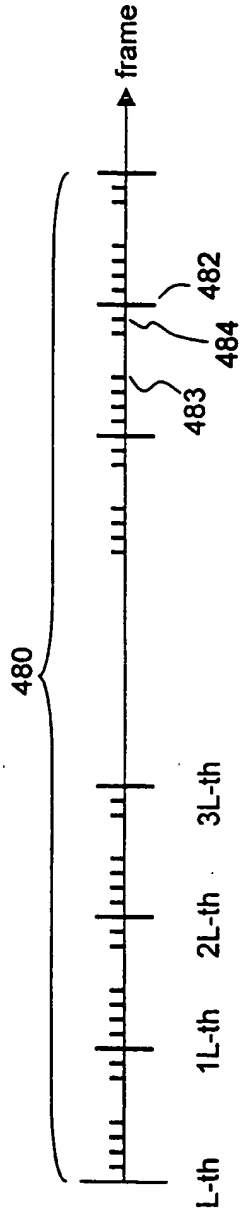


Fig.4F

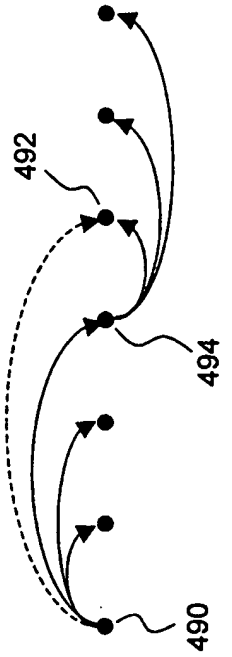


Fig.4G

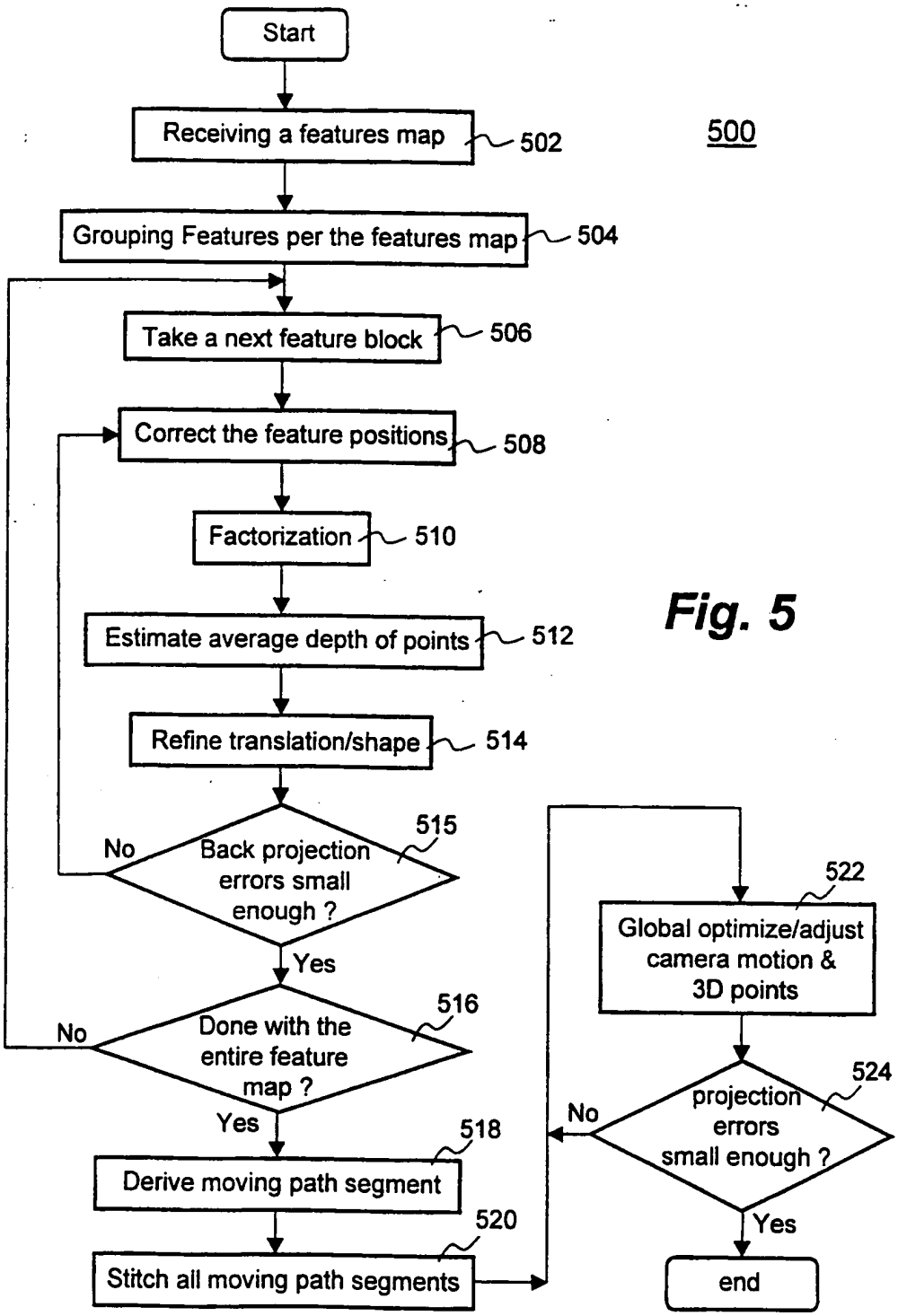
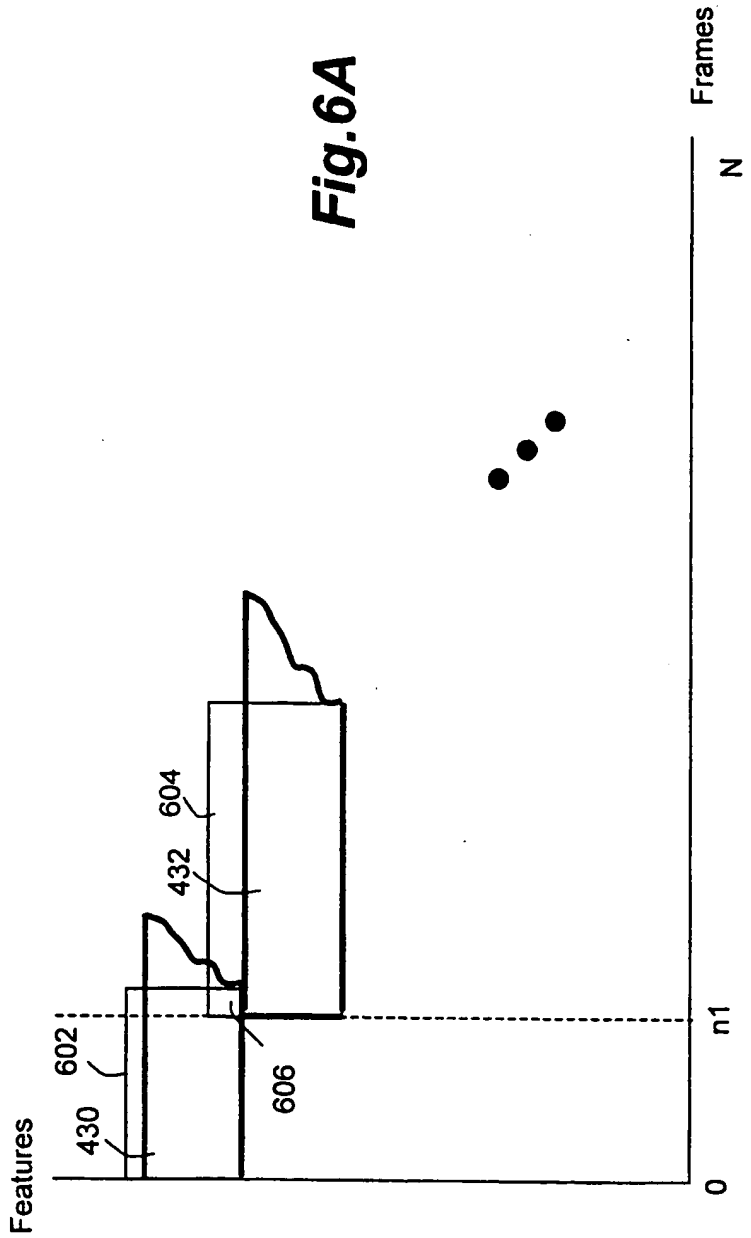

Start
Receiving a features map
502
500
Grouping Features per the features map
504
Take a next feature block
506
Correct the feature positions
508
Factorization
510
Estimate average depth of points
512
Refine translation/shape
514
Back projection errors small enough ?
515
No
Yes
Done with the entire feature map ?
516
No
Yes
Derive moving path segment
518
Stitch all moving path segments
520
Global optimize/adjust camera motion & 3D points
522
projection errors small enough ?
524
No
Yes
end

Fig. 5

Fig.6A



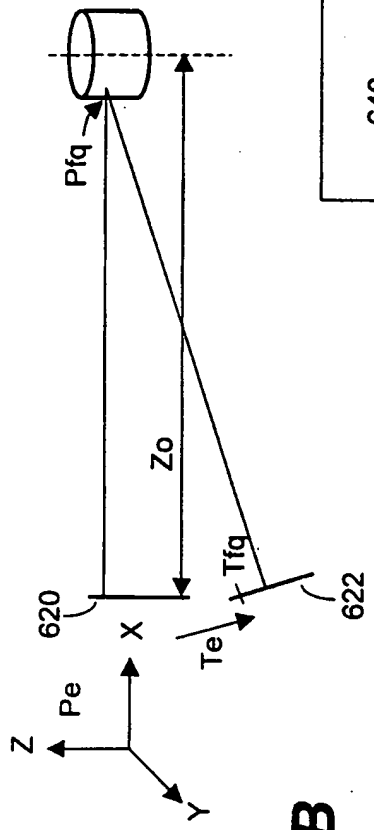

Pfq
Zo
Tfq
620
622
X
Te
Pe
Z
Y

Fig.6B

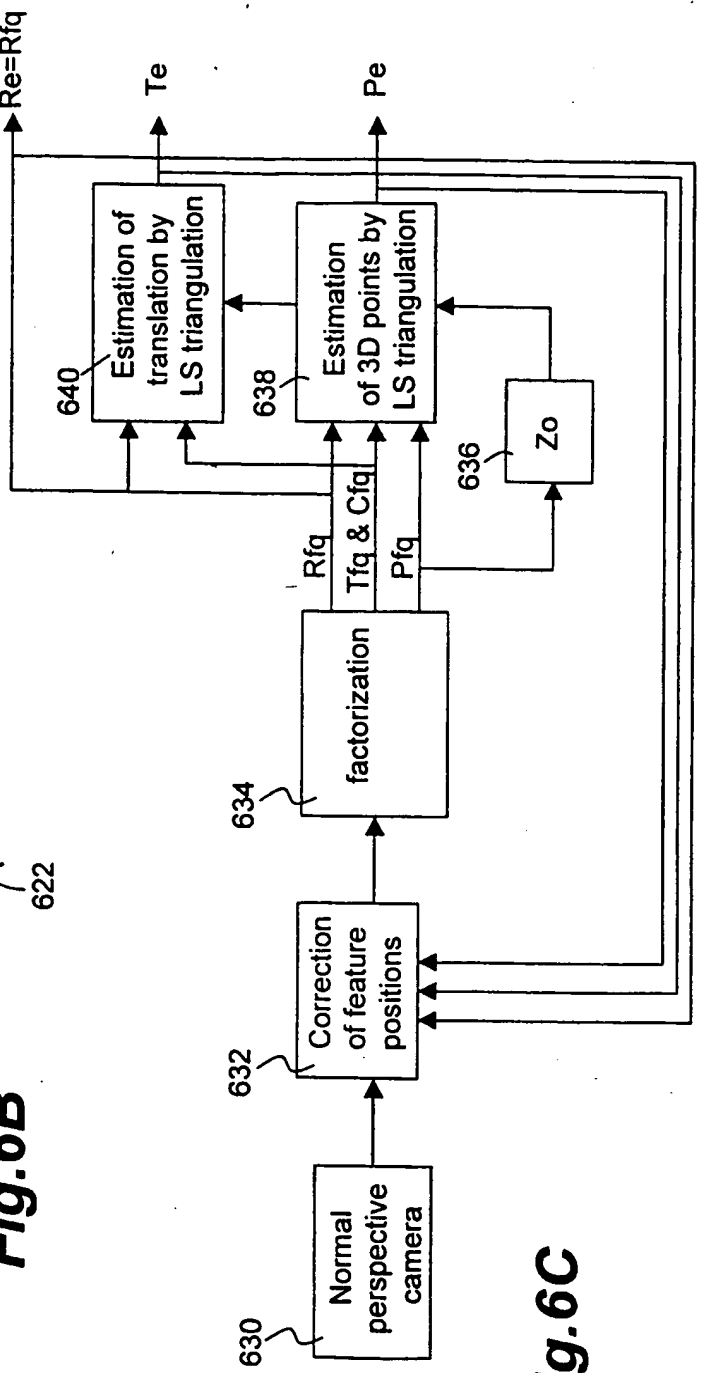

Re=Rfq
Te
Pe
Estimation of translation by LS triangulation
640
Estimation of 3D points by LS triangulation
638
Zo
636
Rfq
Tfq & Cfq
Pfq
factorization
634
Correction of feature positions
632
Normal perspective camera
630

Fig.6C

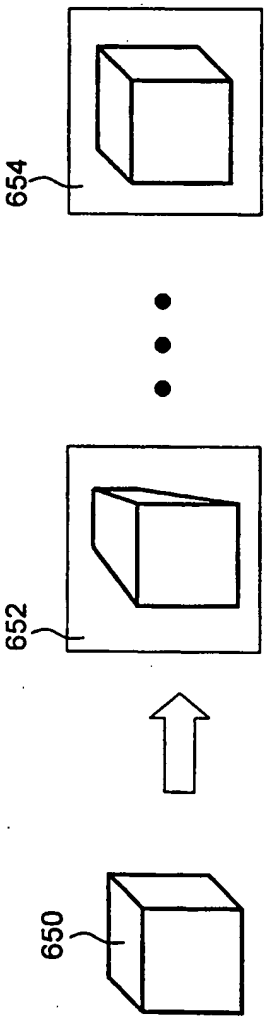


Fig.6D

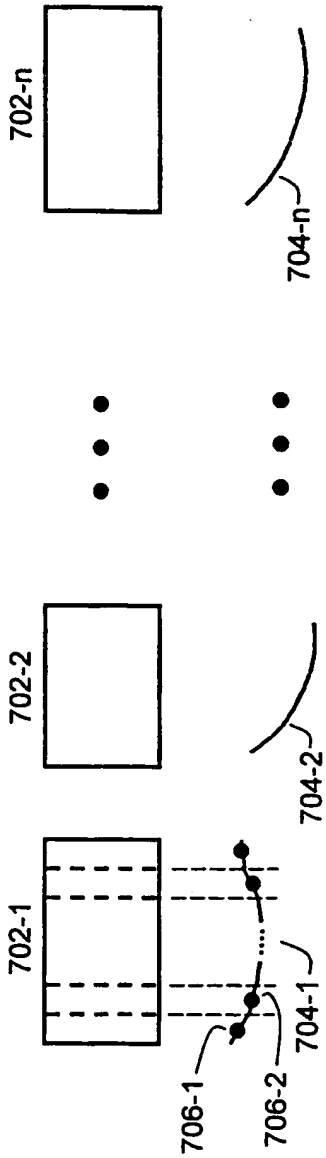


Fig. 7A


712
710
708
704-2
706
704-1

Fig. 7B

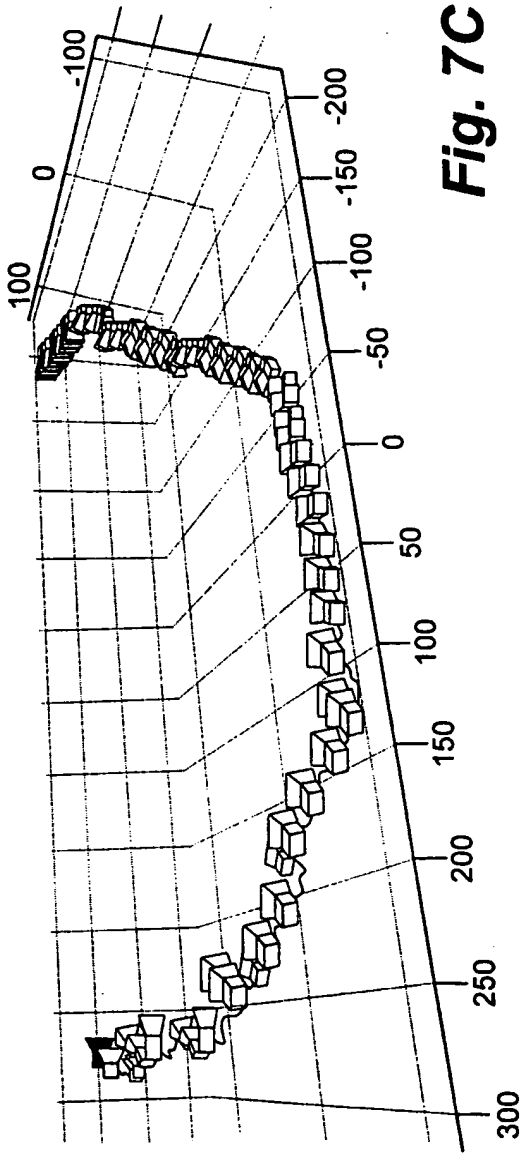

100
0
-100
-200
-150
-100
-50
0
50
100
150
200
250
300

Fig. 7C

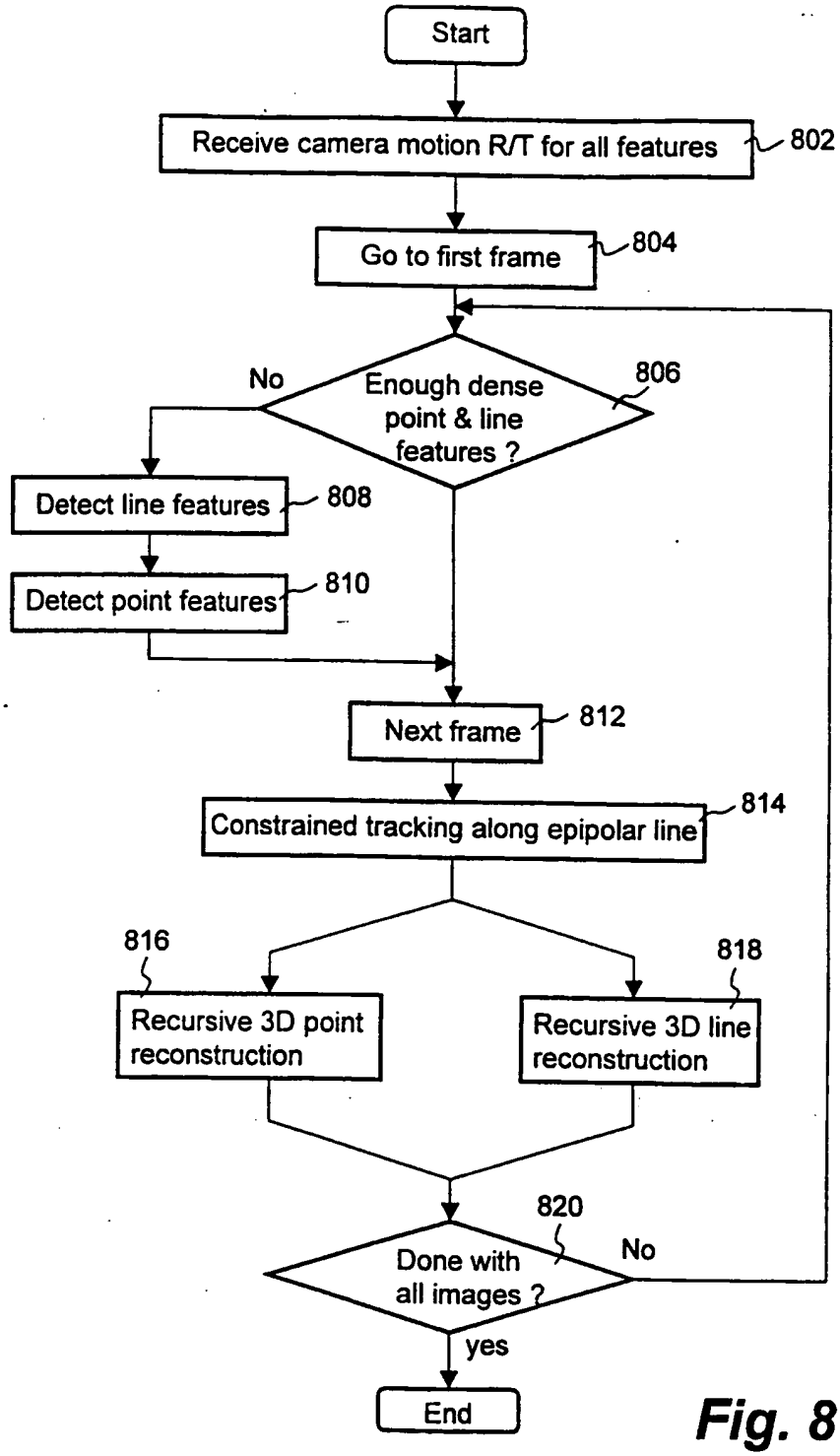

Start
Receive camera motion R/T for all features
802
Go to first frame
804
No
Enough dense point & line features ?
806
Detect line features
808
Detect point features
810
Next frame
812
Constrained tracking along epipolar line
814
816
Recursive 3D point reconstruction
818
Recursive 3D line reconstruction
820
Done with all images ?
No
yes
End

Fig. 8

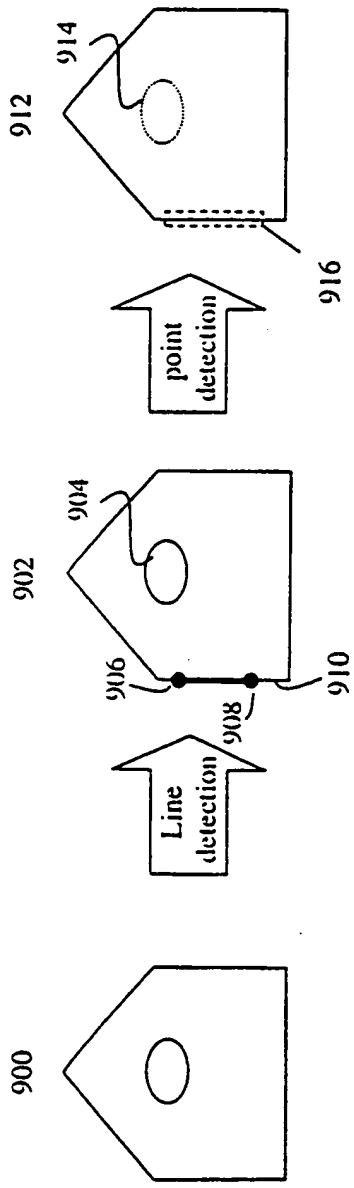

900
902
904
906
908
910
912
914
916
Line detection
point detection

Fig. 9A

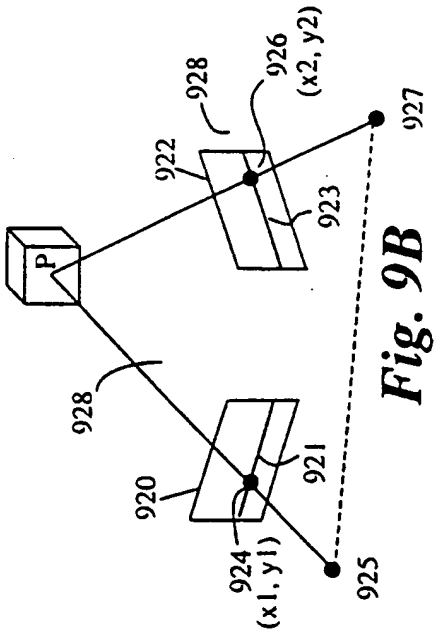

P
920
921
922
923
924
(x1, y1)
925
926
(x2, y2)
927
928
928

Fig. 9B

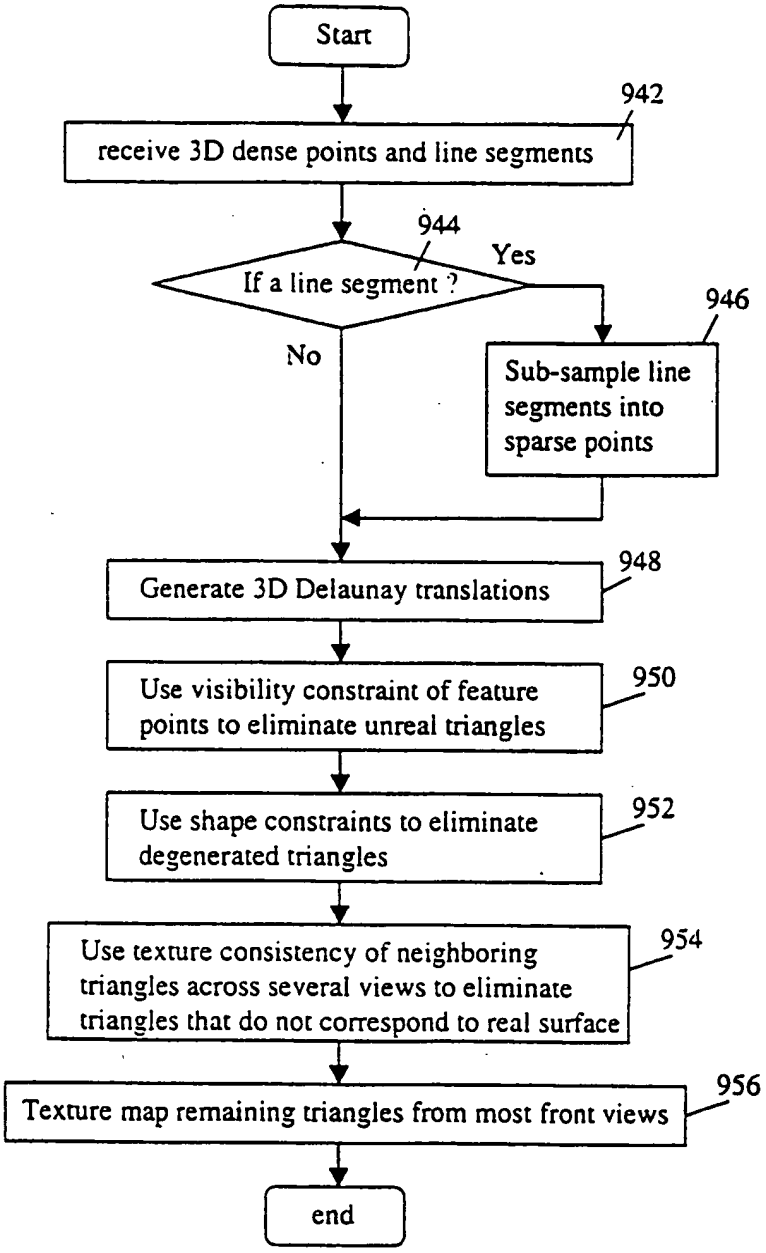

Start
942
receive 3D dense points and line segments
944
If a line segment ?
Yes
No
946
Sub-sample line segments into sparse points
948
Generate 3D Delaunay translations
950
Use visibility constraint of feature points to eliminate unreal triangles
952
Use shape constraints to eliminate degenerated triangles
954
Use texture consistency of neighboring triangles across several views to eliminate triangles that do not correspond to real surface
956
Texture map remaining triangles from most front views
end

Fig. 9C

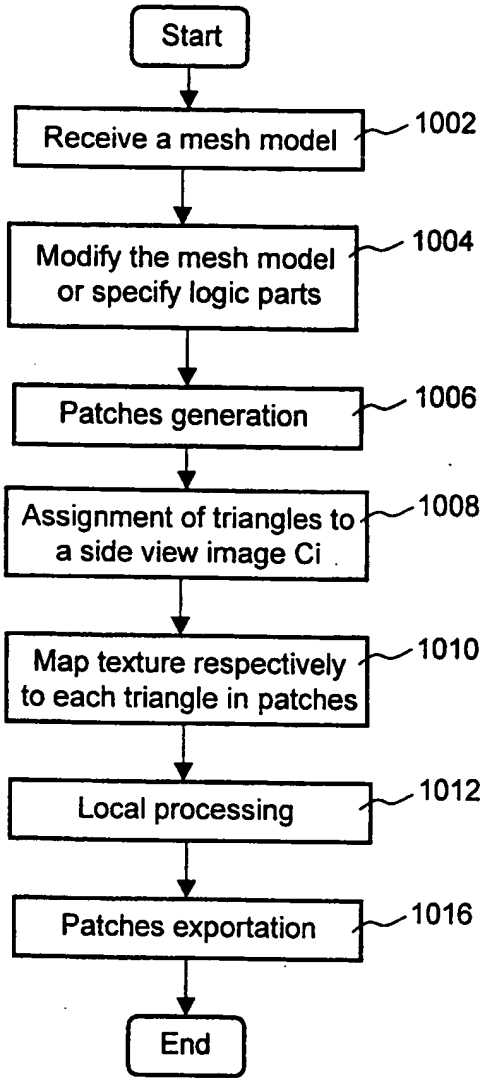

Start
Receive a mesh model
1002
Modify the mesh model or specify logic parts
1004
Patches generation
1006
Assignment of triangles to a side view image Ci
1008
Map texture respectively to each triangle in patches
1010
Local processing
1012
Patches exportation
1016
End

Fig. 10A

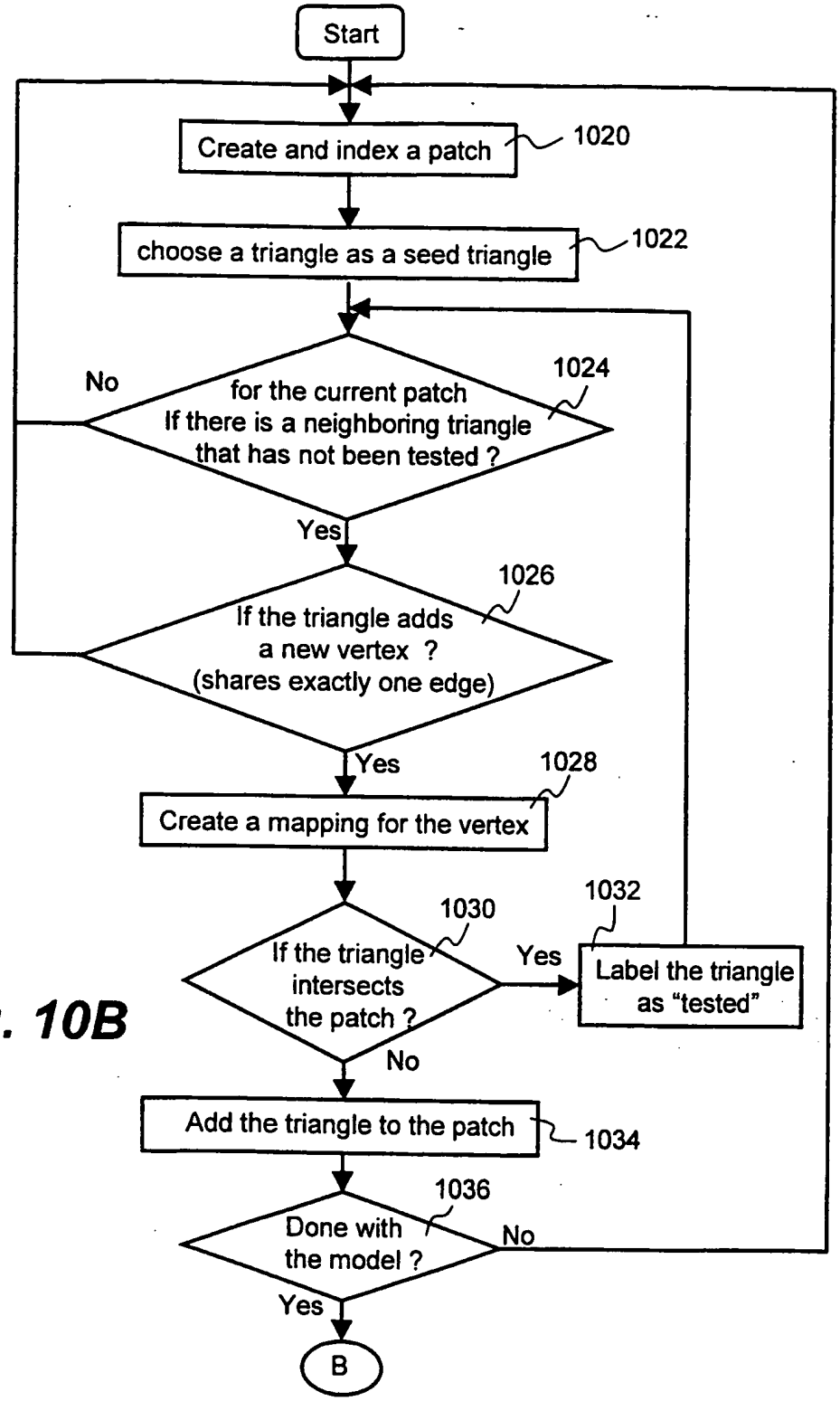

Start
Create and index a patch
1020
choose a triangle as a seed triangle
1022
No
for the current patch
If there is a neighboring triangle
that has not been tested ?
1024
Yes
If the triangle adds
a new vertex ?
(shares exactly one edge)
1026
Yes
Create a mapping for the vertex
1028
If the triangle
intersects
the patch ?
1030
Yes
Label the triangle
as "tested"
1032
No
Add the triangle to the patch
1034
Done with
the model ?
1036
No
Yes
B

Fig. 10B

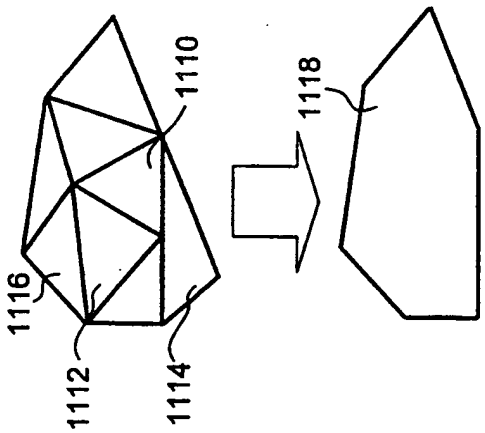


Fig. 11B

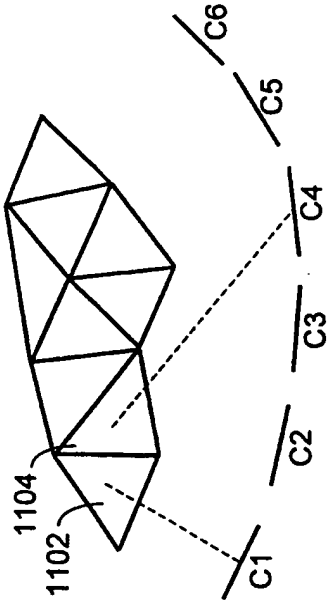


Fig. 11A

INTERNATIONAL SEARCH REPORT

International !lcation No
PCT/US 99/16395

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06T7/20

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06T

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5 511 153 A (ALI AZARBAYEJANI ET AL.) 23 April 1996 (1996-04-23) claims 1-19	1,2
Y	WO 98 17970 A (WAVEWORKX, INC.) 30 April 1998 (1998-04-30) page 4, line 15 - line 25	1,2

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
30 November 1999	06/12/1999

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Chateau, J-P

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 99/16395

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5511153 A	23-04-1996	NONE	
WO 9817970 A	30-04-1998	US 5864640 A	26-01-1999
		AU 5088098 A	15-05-1998
		EP 0934502 A	11-08-1999